NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

UNDERWATER ACOUSTIC NETWORKS: AN ACOUSTIC PROPAGATION MODEL FOR SIMULATION OF UNDERWATER ACOUSTIC NETWORKS

by

Leopoldo Jesús Díaz González Solórzano

December 2005

Thesis Advisor:	Geoffrey Xie
Co-Advisor:	John Gibson

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2005	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Underwater Acoustic Networks: An Acoustic Propagation Model for Simulation of Underwater Acoustic Networks			**5. FUNDING NUMBERS**
6. AUTHOR(S) Leopoldo Jesús Diaz Gonzalez Solórzano			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Underwater Acoustic Networks (UANs) hold enormous potential for both military and civilian applications. However, current networking protocols implemented are often sub-optimal, resulting in severely underutilized networks. We believe one of the key reasons for this shortcoming is a lack of good underwater acoustic propagation models for simulation packages used to evaluate UAN protocols. This thesis addresses this problem by developing a computationally-efficient approximation of a sophisticated analytical model called Monterey-Miami Parabolic Equation model (MMPE). The approximation can then be used to support UAN simulations. The characteristics of the problem make a statistical approach the methodology of choice for this study. Data was generated using the MMPE model. The data was used to develop a much less complex approximation for which an OpNet simulation module could be developed. The latter allows UAN operation to be modeled over a collection of nodes and over an interval of time, rather than a single point in time between two specific nodes, as modeled by the original equation. The result of this research can enable a more complete analysis of network enabling protocols and support more informed decisions regarding the appropriate node topology and protocols to use in order to increase network performance.

14. SUBJECT TERMS Underwater Acoustic Networks, Network Simulation, Propagation Models, Propagation Loss, Link Layer			**15. NUMBER OF PAGES** 113
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

UNDERWATER ACOUSTIC NETWORKS: AN ACOUSTIC PROPAGATION MODEL FOR SIMULATION OF UNDERWATER ACOUSTIC NETWORKS

Leopoldo Jesús Díaz González Solórzano
Lieutenant Commander, Mexican Navy
B.S., Naval Academy, Veracruz 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE
AND
MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2005

Author: Leopoldo Jesús Díaz González Solórzano

Approved by: Geoffrey Xie
Thesis Advisor

John Gibson
Co-Advisor

Peter Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Underwater Acoustic Networks (UANs) hold enormous potential for both military and civilian applications. However, current networking protocols implemented are often sub-optimal, resulting in severely underutilized networks. We believe one of the key reasons for this shortcoming is a lack of good underwater acoustic propagation models for simulation packages used to evaluate UAN protocols.

This thesis addresses this problem by developing a computationally-efficient approximation of a sophisticated analytical model called Monterey-Miami Parabolic Equation model. The approximation can then be used to support UAN simulations. The characteristics of the problem make a statistical approach the methodology of choice for this study. Data was generated using the Monterey-Miami Parabolic Equation model. The data was used to develop a much less complex approximation for which an OpNet simulation module could be developed. The latter allows UAN performance to be modeled over a collection of nodes and over an interval of time, rather than a single point in time between two specific nodes, as modeled by the original equation.

The result of this research can enable a more complete analysis of network enabling protocols and support more informed decisions regarding the appropriate node topology and protocols to use in order to increase network performance.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I dedicate this thesis to my wife, Melanie, my life companion, for her support and encouragement. Without her motivation, this journey would have been much more difficult. My dedication extends to my three children, Alan, Ivan and Ania, who demanded my attention and at the same time gave me the space needed to do my work.

I would like to thank my thesis advisor, Prof. Geoffrey Xie, for his support, encouragement, effort and the challenge he imposed on the process. My gratitude is also extended to my thesis co-advisor Prof. John Gibson for his effort and mentorship.

I thank Prof. Kevin Smith, who provided knowledge and advice about the MMPE propagation model.

I would like to thank the Mexican Navy for giving me the opportunity for this memorable learning experience, and to NPS, professors, students, and staff, for providing such a friendly and gratifying atmosphere.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. CONCEPT

1. Applications

Underwater sound propagation has been used either for military applications like sonar, mine fields, voice communication, or civilian use such as hydrographic surveys, oceanographic studies, and marine life research.

Wireless communications to this date are a common part in our daily life, and the term wireless is usually associated with over the air communications and not related to underwater communications. Underwater networks may use radio, sound, light, or a wire as the mean to transport data from one point to another. Radio would require long antennas (due to the low frequency used) and consume a tremendous amount of energy. Light can only be used at very short ranges while wired networks present the inconvenience of been fixed or semi-mobile. In comparison, sound propagates on the water in a more efficient manner [Urick 1983].

Underwater acoustic networks (UANs) are wireless networks that typically consist of multiple sensor nodes and one or more relay nodes or gateways. The sensor nodes gather information and transmit data, either directly or through a sequence of relays, to a gateway, which then forwards the data to other networks or shipboard data processing centers.

Underwater acoustic networks make rapidly deployable and highly movable underwater sensor grids possible. They can also be used to navigate and control UUV's to perform tasks such as mapping an ocean contour or exploring a shipwreck. One of the key factors in the exploration business is the care of the "umbilical cord" that can be cut by or tangled with the wreck structure.

A UAN sensor network can also give warnings of earthquakes or tsunamis. The use of cable (wired networks) may not be suitable in certain areas. For the study of internal currents and eddies where the sensor nodes have to drift, the use of acoustic communication instead of retrieving the nodes for post analysis, could provide a better

understanding of those currents in a close to real time fashion. Other possible applications of UANs include environmental monitoring and track pollutants or other substances of interest.

In the tactical arena, the UANs can be used to provide surveillance. Because they does not rely on a cable (which would represent a single point of failure) to operate, they are more resistant to attacks.

2. Acoustic Propagation Problem

Sound propagation in the ocean water is roughly 20000 times lower than the radio channel, between 1450 and 1540 meters per second [Lysanov 1982]. Even transferring a message over a moderate distance, say 500 meters, would incur a significant amount of propagation delay. Moreover, there are random periods of link discontinuity in which no reception is possible Therefore, UANs can be placed in the category of Delay Tolerant Networks [Gibson 2005] for which traditional networking protocols may not work well and new approaches are being developed, to address the characteristics of such networks.

Underwater sound propagation depends on the characteristics of the water column that a particular ray of sound travels by, and any changes on the medium will cause that ray to travel in a different path, or at a different speed, no to mention that the path and absorption of a sound wave depends also on the frequency of the signal. Diverse frequencies will have singular deviations of the travel path that can impact the reception or not of an acoustic signal.

The problem of extreme propagation delays is compounded by the limited bandwidth of the communication medium, typically in the order of one kilobit per second (Kbps) [Xie 2001]. To address this, problems new approaches and protocols need to be developed.

B. CHALLENGES

1. Previous Efforts

The work of this thesis is motivated by a previous study, which compares he UAN performance under two types of bandwidth allocation protocols: contention based and priority based [Gibson 2005] and a thesis developed by a NPS graduate, which

evaluates the impact of media access control on the UAN performance [Coelho 2005]. The author actively participated in the latter effort. The author had also worked on an oceanographic ship and experienced the hardship of manually retrieving information from individual underwater sensors. It would be much easier if an Unmanned Undersea Vehicle (UUV) could be deployed to download the data automatically from the sensors via a UAN.

UAN has been the subject of recent studies and computer simulation models have been created to analyze the behaviors of such networks. The model in [Coelho 2005] was used to compare the performance of a UAN under a contention based with collision avoidance protocol and a contention free (aloha like) protocol. The model in [Sozer 1999] is derived from the OpNet radio pipeline stages. One significant shortcoming of these two models is their lack of a realistic acoustic propagation model. The Coelho model does not consider any propagation loss, and the Sozer model does not consider the depth of the sensors, and uses a formula for propagation loss that is best suited for low frequencies (100 Hz-3 KHz) [Lysanov 1982]

Physicists have developed complex propagation loss models that for a given area and environmental characteristics provide high level of accuracy [Smith 1999], these models are focused on obtain a description of the sound propagation particularities over the area of interest, but the tradeoff is the increased amount of time that takes to obtain a propagation loss figure.

2. Remaining Problem

The UAN simulation propagation loss model are simple to implement but does not necessary reflect the reality of the sound propagation loss.

The Physicist approach although is very accurate, are to complex to be implemented in a simulation.

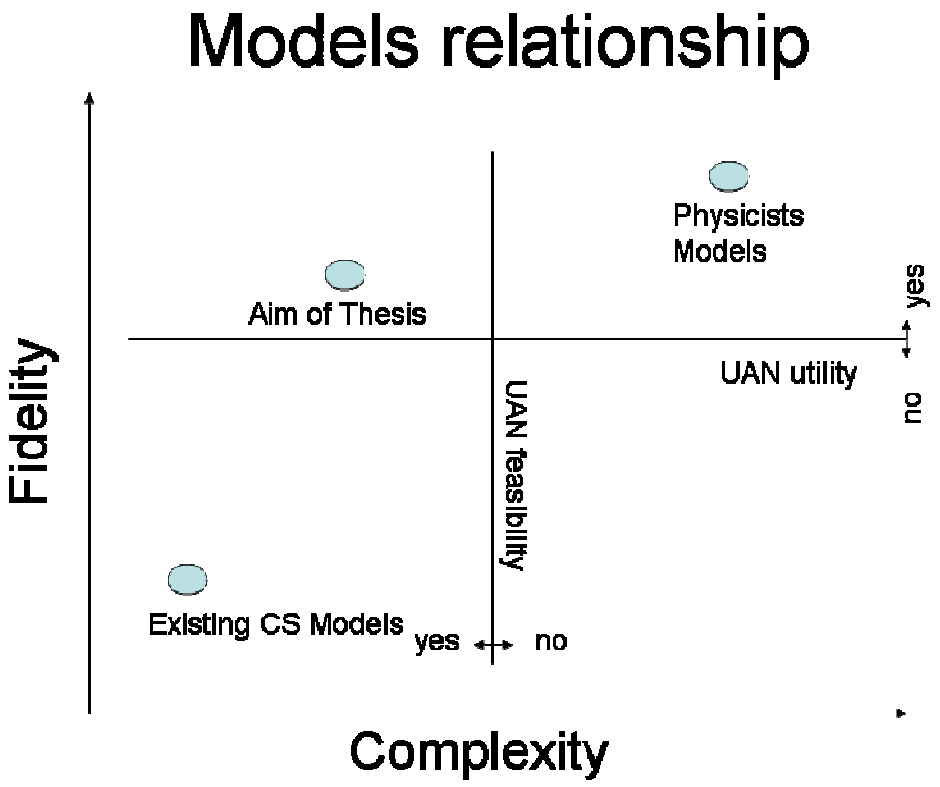

Figure 1. Model Practicality.

3. Problem Statement

While the Physicist model provides an accurate calculation of the delay at a given point in time, network modeling requires near continuous references to propagation delays, the volume of which makes it computationally infeasible to support using the complex physicist model. Thus, it is necessary to derive an approximation to the physicists' model that supports the time constraints of the network simulation while preserving some of the fidelity of the former.

The objective of this thesis is to develop a computational reasonable approximation of one physics model, The Monterey-Miami parabolic equation model (MMPE),described in chapter two was selected because is well documented, well established [Smith 1999] and we have access to the source code, approximation that can be integrated with existing UAN models to have a better simulation results.

The obvious research questions to answer are:

Can we incorporate a physics model directly to a simulation?

Can we make an approximation that combines the benefits of the simulation and the physicist model?

This thesis will provide an approximation to the MMPE model, that is suitable for computer simulation, and then integrate that propagation loss model as a module to a UAN simulation, and see how this impact the simulation without the propagation loss module.

C. THESIS OUTLINE

This thesis begins with a description of the underwater acoustic theory, two simulation propagation loss models that has been proposed for UANs, the OpNet propagation radio transceiver propagation loss model for illustrative purposes and a detailed (not theoretical) description of MMPE simulation model, Chapter III describes the process to obtain data and develop an approximation to the MMPE model. Chapter IV provides a description of the simulation modifications. Chapter V presents the simulation results and presents some pertinent conclusions and recommendations for follow-up work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND AND RELATED WORK

A. INTRODUCTION

This chapter first introduces an underwater acoustic history summary and basic acoustic theory; later the chapter describes two UAN simulations models, the OpNet radio transceiver pipeline propagation model, and then describes the Monterey-Miami parabolic equation model, after the description the chapter provides with the conclusions about the particularities of the exposed models.

1. Underwater Acoustic History

Leonardo da Vinci in 1490 wrote that "if you cause your ship to stop, and place the head of a long tube in the water and place the outer extremity to your ear, you will hear ships at a great distance from you" [Bell 1962].

It is believed that Daniel Colladon made the first measurement of sound propagation occurred in Lake Geneva in Switzerland circa 1827, when the first measurements of sound speed were made, at the beginning of the 20th Century there underwater sound was used in navigation by means of a submerged bell that sound at the same time that a fog signal the time difference will give a ship the approximately distance to the lightship, by 1914 there were devices that could detect icebergs at two miles of distance. By the end of the second decade of last century a paper on underwater sound was published by H. Lichte in Germany, this paper states the impact that small variations on salinity and temperature have on the path that a sound ray travels trough the water column, after that there were advances in sonar and mine technologies due to the political climate of the post WWI and the pressure of the WWII and the submarine threat, to ally convoys. The use of sound for communications was limited to underwater telephone to provide voice communications between ships and/or submarines, [Urick 1983], but the interest in UAN networks is more recent.

2. Underwater Acoustic Theory

Sound as described by Urick [Urick 1983] is a regular molecular motion in an elastic substance propagating to adjacent particles. The mechanical energy that is transmitted from particle to particle from the source and propagates through the ocean at the sound speed constitutes a sound wave. As Lichte stated the propagation of such

waves will refract upward or downward according with changes in salinity, temperature and pressure. This is change is also related to sound velocity that travels from 1450 to 1540 meters per second and present small variations but this small changes have a great impact to the propagation of sound [Lysanov 1982].

These sound waves are “bended” downwards if the sound velocity decreases monotonically [Lysanov 1982].

Transmission loss (TL) is defined as the decrease in sound intensity from a location one unit of distance from the source to a distant location in the ocean. Among the sources of this data loss are attenuation and spreading the previous is range dependant and is a combination of absorption and scattering, Magnesium Sulfate ($MgSO_4$) relaxation is the major source for absorption, below 100 KHz. attenuation usually is expressed in decibels per kilometer. The later is a geometrical effect due to the sound wave propagating from the source this value varies logarithmically with the range [Lysanov 1982] [Urick 1983].

Diverse empirical formulas has been developed trying to account for attenuation loss like the one proposed by Thorp [Urick 1983] [Lysanov 1982].

$$\alpha = \frac{0.11 f^2}{1 + f^2} + \frac{44 f^2}{4100 + f^2} [dB / Km]$$

And for spherical spreading as:

$$SS = 20 \log r$$

Then the following formula can be used to obtain the signal propagation loss, denoted by *loss*:

$$TL = 20 \log r + \alpha r \text{ x } 10^{-3}$$

Where r is the range in meters f is the frequency in KHz. and α the attenuation coefficient.

The attenuation coefficient obtained with this formula is described more realistically at frequencies between 100 Hz and 3 KHz [Lysanov 1982].

There are other models to obtain propagation loss that have the main purpose of solving the wave equation for a particular problem, this equation is a partial differential equation that considers sound velocity c, the acoustic pressure p, a coordinate system x, y, z and the time t. The equation is of the form: [Urick 1982]

$$\frac{\partial^2 p}{\partial t^2} = c^2\left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2}\right)$$

Two approaches has been proposed the normal mode theory and the ray theory , the previous, is known as normal modes , and consist of a series of characteristic functions that are added to fulfill specific conditions, the model result in a complicated mathematical function, that provide less insight than the ray theory. Ray theory that utilize rays that reassemble optic theory , this creates a ray diagram , that can provide a better visualization of the sound paths , this model is limited because the results are not as accurate if the radius of curvature or the pressure amplitude changes largely over one wavelength [Urick 1983].

B. COMPUTER SCIENCE MODELS

1. Design and Simulation of an Underwater Acoustic Local Area Network [Sozer 1999]

This model design and tested a UAN by means of the OpNet Radio model assuming stationary nodes, some of them are consider as sensor nodes and one as the master node that gathers the data from the other nodes and transfer that data to another instance. Nodes are grouped into clusters that have a unique frequency band.

On the power module, this model incorporates a modified OpNet radio model; the path loss for signal of frequency f is described by the formula:

$$loss = d^{\gamma} \exp(d\alpha / 10)$$

$$\alpha = \frac{0.11 f^2}{1 + f^2} + \frac{44 f^2}{4100 + f^2} + 2.75 \text{ x } 10^{-4} f^2 + .003$$

Where d stands for the propagation distance, γ is the path loss exponent set to two in this case, this model also define a Rayleigh random variable to account for fading., and a background noise level that reaches 20 dB at maximum range [Sozer 1999].

The formula is of the Thorp form [Urick 1983], and as stated previously the first two terms are best suited for signals in the 100 Hz to 3 KHz range. This model does not account for depth yielding a network that stays in one plane with all nodes at the same depth (not considered), for instance a node at 900 meters but at 12 meters depth will receive the same figure for propagation loss than a node at the same distance located at 300 meters depth, on the other hand this model does tries to represent fading.

2. Evaluation of the Impact of Media Access Control on Latency, in a Delay Constrained Network [Coelho 2005]

This model represents the performance of two media access control protocols, (MAC), a carrier sense multiple access with collision avoidance (CSMA/CA), and an Aloha-like protocol.

The carrier sense multiple access protocol with collision avoidance sequence is as follows the sender will first wait for a Distributed Inter Frame Space (DIFS), the make a request to send (RTS) the receiver then waits for a Short Inter-frame Space (SIFS) and answers a clear to send (CTS), the sender will now send the data with a SIFS between each data packet after the last packet is received the receiver waits for a SIFS and answers with an acknowledge (ACK).

The aloha like protocol sequence performs as follows the sender sends data to the receiver as the data is available to the sender, the receiver node will acknowledge the frames received after wait for a SIFS, the sender will time out with a random back-off and then retransmit if it does not receive an acknowledgement for the data just send (the sender will ignore any other acknowledge but the one it is waiting for) once the ACK is received the sender performs a Back-off and the process is repeated for each frame, the relay node will wait until all frames from a message , this is a complete message, is on that relay node to stat the retransmission of the message to the next node in the transmission path.

The model states that does not account for propagation loss and only accounts for the behavior of the mentioned link layer protocols, the model determines the receiving nodes in a binary fashion, all nodes at range less or equal than 1500 meters from the sender will receive the transmission with out error or losses, depth is also not considered [Coelho 2005].

3. OpNet: Radio Transceiver Pipeline

OpNet in the wireless Manual states a Received Power Model (dra_power), which calculates the intensity of a given signal when reaching the receiver, for calculating the power the model takes the base frequency and the bandwidth to calculate the center frequency, then calculates the wavelength (lambda) via dividing the speed of light by the center frequency, then using the propagation distance OpNet calculates the free space propagation loss as defined by:

$$loss = \left(\frac{\lambda}{4\pi d} \right)^2$$

Where *loss* stands for propagation loss, λ the wavelength, d the range or distance between nodes [OpNet 2003].

The result from this equation is placed on a variable called path_loss in the case that the distance or range is set to zero the variable is set to one.

The model then takes the transmitted power, receiver antenna gain, transmitter antenna gain, and multiply those terms with the path loss to obtain the received power figure [OpNet 2003].

C. PHYSICIST MODEL

1. Monterrey-Miami Parabolic Equation (MMPE) [Smith 1999]

The MMPE model is used for predicting underwater acoustic propagation, utilizing a parabolic equation that is an approximation to the full Helmholzt equation (wave equation), this equation is based on a split-step Fourier algorithm , the sound pressure is calculates in small increments/variations in range and depth, forming a grid, if we increase the increment size, we can obtain a better performance but the best solutions are found to be when the increments lie within small wavelengths: maximum 2λ in range and 4λ in depth, the parabolic equation (PE) is a full wave and includes all wave effects like diffraction, this model utilizes an hyperbolic mixing function to address the sound speed discontinuity, the scale for this function by experience is found to be $\lambda/10$, (this is to model a step function discontinuity), but smaller values are found to degrade

the solution, for density discontinuity the model uses a cubic spline function the optimal mixing length for this is 2λ. For Range the optimal solution was found to be when an increment of approximately one λ, and if the range step decreases more the propagation function accuracy will be reduced [Smith 1999].

a. Considerations

The model assumes all the surfaces are perfect reflectors; it considers the bottom as a fluid with specific characteristics, to account for sediment layers effects.

b. Validation

The parabolic equation method was introduced by Professor Frederick Tappert in 1977, and later a University of Miami Parabolic Equation Model (UMPE) was developed in 1993 by Smith and Tappert, when Professor Kevin Smith moved to NPS an improvement to the model was made and the new model was denominated the Monterey-Miami Parabolic Equation Model.

The model was validated and compared to known solutions at the Shallow Water Acoustic Modeling Workshop (SWAM'99), by professor Kevin Smith [Smith 1999] and the details of the comparison can be found in the referenced article.

c. Implementation

The model is implemented in FORTRAN has a command line user interface, and receives its inputs trough files that describe the environment, and the characteristics of the sound source. This program produces a binary file that needs to be interpreted with two Matlab programs for post processing; hence, the model is stand alone but needs Matlab to finish and mange the information.

The program relies on the following text files [Tappert MMPE]:

- pefiles.inp : this is the main file that contains the name of the other files that describe the environmental characteristics, and have the input arguments(as lines in the text file):
- nzout, depmin[m], depmax[m]: number of points in depth, the minimum and maximum depth respectively.
- nrout, rngmin [km], rngmax [km]: number of points in range minimum and maximum range.
- The previous two rows are considered as "requested values" and the model will try to get as close as possible to the proportioned inputs.

- Nz, dr [km], depcal[m], c0 [m/s]: stands for the vertical FFT size (integer multiple of 2!), range step, maximum depth of calculation, and reference sound speed.
- On the last row all the values but c0 can be set to zero which is always 1500 m/s. default values are used that proportioned the most accurate result, to obtain more efficiency the values can be modified to decrease run times by making compromises.
- pessp.inp : this file holds the sound speed profile organized in rows of data, the first row contains two numbers that represent (from left to right) the number of radials in azimuth, and the total azimuth distance represented by the radials, (in this version of the MMPE only the first radial is used),the second row indicates the number of speed profiles ,in the first radial, the third row contains the range (Km)of the profile (how far from the source), and the number of speed profiles at that range(the number of sound speed changes at that range), the next rows (according to the number of speed profiles , one by row) contains the depth (m) and the sound speed figure at that depth (m/s).
- pebath.inp this hold the bottom profile , the first row contains the number of radials in azimuth and the azimuth aperture (not used, in this version), the second row contains the quantity of bathymetric points defined, the third row holds the range (Km) and the depth (m) of the point(one row per point).
- pebotprop.inp this file is used to specify the properties of the sediment interface the first line contains the number of points where the profile is defined, the following lines (as much as the number defined by the first line), contains (in sequence)the range (distance of the source),sound speed, sound speed gradient (1/s), density (g/cm^3), compressional attenuation(dB/m/kHz),shear speed(m/s),shear attenuation(dB/m/kHz).
- pedbotprop.inp the file has the properties of the deep layer and the format is similar to the pebotprop.inp.
- pesrc.inp this file holds the characteristics of the sound source , two kind of sources are allowed a vertical line array and a wide angle punctual source, the first row specifies the source depth, the second the array length, the third the steering angle of the main beam of the array (degrees), positive steer the beam down, negative upwards. The third stand for the center frequency (Hz), the fourth the bandwidth (Hz) and the fifth the number of frequencies (must be a power of two).

The program then will calculate the propagation loss for the conditions specified on the previously mentioned files and then provides a binary file as an output, the file then can be post processed using two Matlab programs.

(1) Post-processing the Matlab files that the model utilizes for post-processing are peout.m and peout2.m.

peout1.m, which after prompting for the MMPE output file name, reads the file and places a set of variables on Matlab workspace (serves as initialization, and it is only needed to run once), and peout2.m that post process the file (this file can be run several times to obtain the desired data).

The post processing file peout2.m has a menu with six options:

Output starting field data;

Compute data for a single radial;

Compute data for a single range;

Compute data for a single depth;

Compute data for a single interface;

Compute travel time data;

The user then selects the desired option and obtains the data of interest, that is presented in a graphical manner (the information can be saved as an image or as a data file.)

d. Performance

The model is computational intensive and the performance is frequency dependant, for instance in a 2.4 Gigahertz (AMD 3700+) laptop one run of the model at 20 kHz, with 2000 range steps, and 600 depth steps, running at the real time (windows XP) priority, took approximately 30 minutes, the same model at 1 KHz consume around 15 seconds but at 400 Hertz, it took approximately three seconds to produce an output file, these last two frequencies although take less time to compute are not within the range of interest to our model.

D. CHAPTER CONCLUSIONS

There have been many attempts for modeling the behavior of the underwater acoustic networks and some of the model tried to model the physical layer characteristics, using a empiric formula like the Thorp formula may give a good approximation but those models does not account for depth variations, the OpNet propagation model (even it is for radio waves) does not account for changes on depth (altitude), but that maybe due to the

directional behavior of radio waves, the MMPE propagation model gives an accurate approximation of the propagation loss, but at the frequencies that are required by the UAN's to transfer data , the model takes too much time to achieve a solution, and requires Matlab for off line post-processing.

For example at 20 kHz a single transmission event, given a 20 node network would require nearly 600 minutes to compute the reception likelihood and delays to every node in the network. Assuming that the average propagation delay is 750 ms, the calculated values are five orders of magnitude too slow to support real-time simulation. Given a transmission event for each node once per minute (a low traffic scenario), that represents 200 events for a 10-minute scenario, not including response traffic. This is 6000 minutes just to calculate the delays. Clearly, is not possible that this model can support a network simulation.

The MMPE model is suitable for providing the data that can be analyzed to produce an approximation to the model.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DEVELOPMENT OF AN APPROXIMATION MODEL

A. INTRODUCTION

This chapter first introduces an overview of the approaches taken to obtain a propagation loss model, using the MMPE model as a base. Then it discusses the steps taken to obtain propagation loss data, which can be used to generate an approximation for the MMPE model. Later, the chapter presents the analysis of the data generated by the model equations and introduces a wave function to model the effect of the wave movement in the ocean. Finally, it introduces a random function to model background noise.

1. Overview of Approaches

a. Direct Manipulation of MMPE Model

Our first approach, since we have access to the source code, was to simplify the mathematics of the MMPE model directly.

However, we found that in order to compute a propagation loss figure the MMPE model implementation works in an iterative way and needs the previous result to compute along a given range. In the "coarse" mode, the model makes a calculation grid of three wavelengths in range and half of a wavelength in depth, [Smith 1999]. At 100 meters depth, 1 kilometer distance, and a frequency of 20 kHz, the model requires one calculation each 0.225 m in distance and 0.0375 m in depth, for a calculation grid of 4444 by 2666: close to 12,000,000 iterations.

If we increase the depth increment above one tenth of wavelength the solution begin to degrade [Smith 1999]. In the "coarse" mode, the depth increment is already four times more than that limit. Moreover, because of its iterative nature, the model needs to calculate the whole area to compute the propagation loss between two points, and cannot directly compute the propagation loss of a specific transmission path given the positions of the sender and receiver. Additionally, in order to get the desired information, the output of the model needs to be post processed.

We concluded that direct manipulation of the MMPE model cannot reduce its complexity sufficiently to a level that is sufficient for network simulation.

b. Statistical Approximation of MMPE Model

Our second approach was to develop a statistical approximation using nonlinear regression to the MMPE model. This approach produced a satisfactory approximation. The main steps for this method are illustrated in Figure 2. First, the original MMPE model is used to produce a detailed propagation loss data map for the target area. Second, the data is smoothed in preparation for statistical analysis. Third, regression techniques are used to obtain an approximation of the propagation loss model from the smoothed dataset. Finally, additional terms are added to account for effect of wave motion and random noises. The result is a function that returns the propagation loss of a transmission path based on a small set of parameters pertinent to the locations of the sender and receiver and the transmission frequency. In the reminder of this chapter, we will discuss these steps in detail.

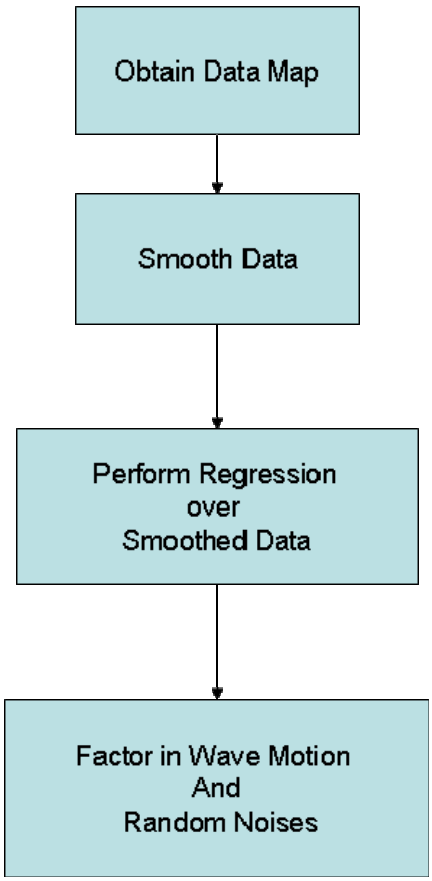

Figure 2. Steps of Statistical Approximation Approach.

B. DATA COLLECTION

1. Model Input

The ocean sound propagation is affected by the physical environment and each area will have its own propagation loss signature. For this reason, a general solution for the physical propagation loss is not intended in this work. Rather, the intent is to show a process that can be followed to make an approximation for an area of interest. To demonstrate the process and make an example approximation for a given area, MMPE model input files were configured for an hypothetical area of one kilometer in range and 250 meters depth with a gradient of 50 meters/Km. The water column reference speed is 1500 m/s from 0 to 100 m in depth. The source depth was varied from 5 to 35 meters at 5 meter intervals, and the frequencies modeled ranged from 10 kHz to 20 KHz, in 1 KHz increments. These values were entered in the MMPE model's "pesrc" configuration files, as shown in Figure 3. One configuration file is required for each source depth, and one for each center frequency modeled. Since we modeled eleven frequencies and seven source depths, we had to perform seventy-seven runs of the model.

```
Source depth [m]          :25.
Array length [m]          :0.
D/E angle [deg]           :0.
Center frequency [Hz]     :10000.0
Frequency bandwidth [Hz]  :1000.
Number of frequencies     :1
```

Figure 3. Sample "pesrc" Configuration Used to Collected Data.

The area that the model uses to compute the data is a slice (radial) of the water column in one direction with a range magnitude and a depth as show in Figure 4. The area more specifically is of trapezoid shape with the particularities showed in Figure 5. Note that the MMPE model has additional configuration options such as a sea bottom profile and a sound speed profile. (See Chapter II for details.) For the purposes of this study, we assume that the sea bottom is a linear slope ranging from 250 m to 300 m in depth and the sound speed remains constant across the water column. Only the frequency and source depth parameters are changed for each model computation.

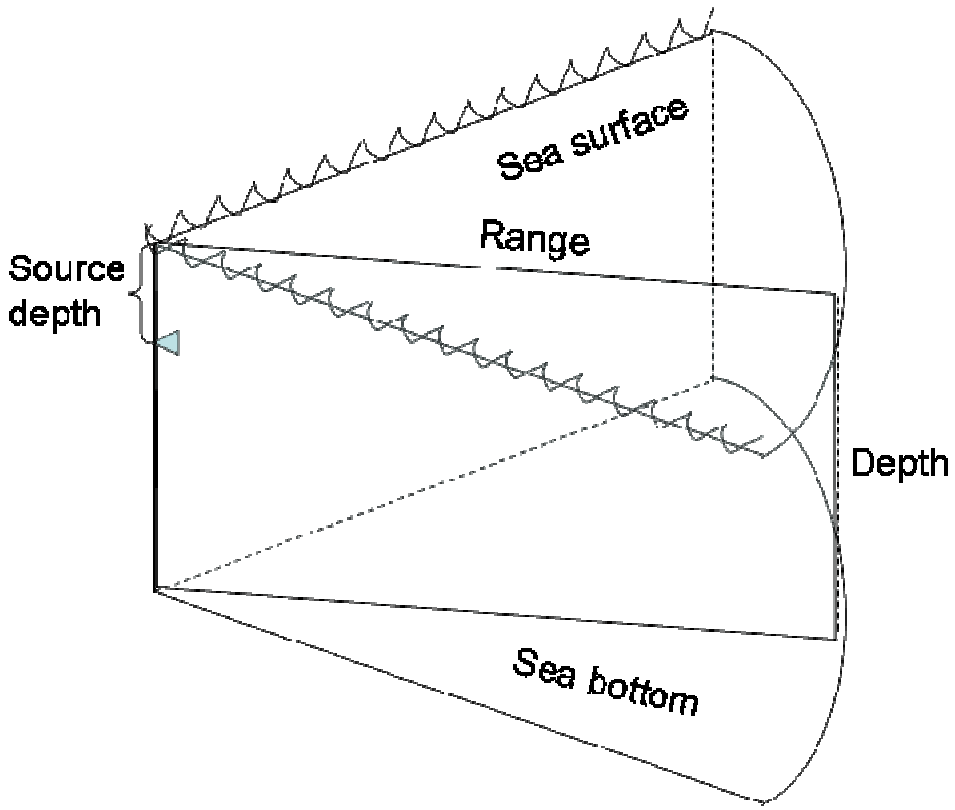

Figure 4. Typical Area Computed by MMPE Model.

Each input was configured by selecting a source depth and the model was run for the eleven frequencies. Then another source depth was configured and the process repeated until all source depths were modeled.

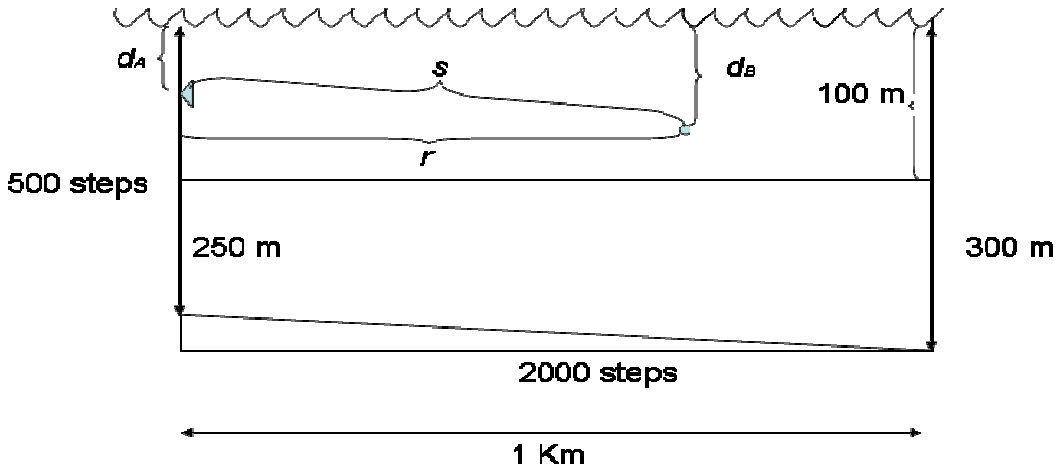

Figure 5. Area Configuration Parameters. r represents the horizontal distance between sender and receiver; s represents the distance between sender and receiver; d_A and d_B represent source depth and receiver depth, respectively.

2. Model Execution

The MMPE model runs from a command line interface, as show in Figure 7. The user has to choose between two options: the efficiency mode and the accuracy mode of calculation. For the purposes of this work, the accuracy option was selected for each of the runs, which means that the depth grid is one tenth of a wavelength and the range grid is one wavelength [Smith 1999]. To execute the 77 runs the model took approximately 20 minutes on a laptop with a 64-bit AMD 3700 processor at 2.4 Gigahertz.

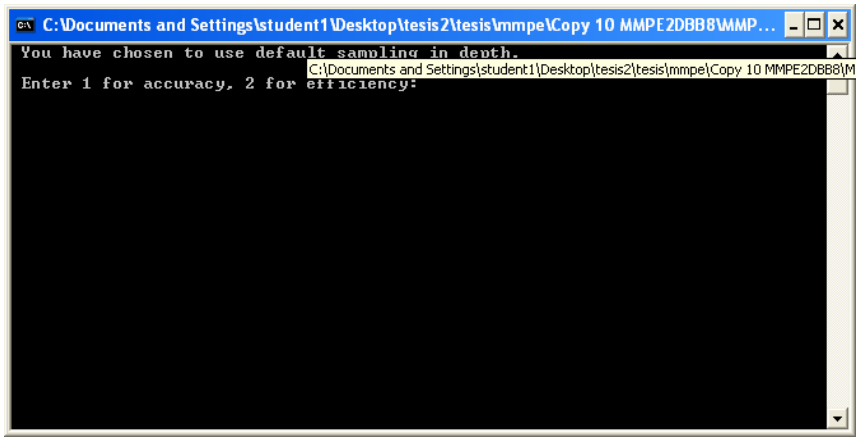

Figure 6. MMPE Interface.

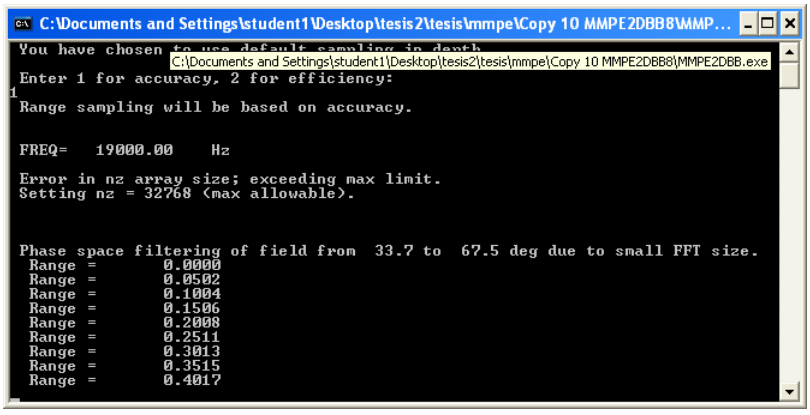

Figure 7. MMPE Interface.

The model presents a screen output showing the range step to indicate progress of the calculation. The output of the model is a binary file that contains the information needed for post processing. The information contained in the output file is preceded by a header that provides specifics about such items as frequency, reference sound speed, number of range points, range step size, source depth, maximum and minimum range for

output, number of depth points, and minimum and maximum depth. The information is organized by frequency, then by range, and finally, by radial for compatibility with 3D versions of the model as shown in Figure 8. [Yonghoon 2000].

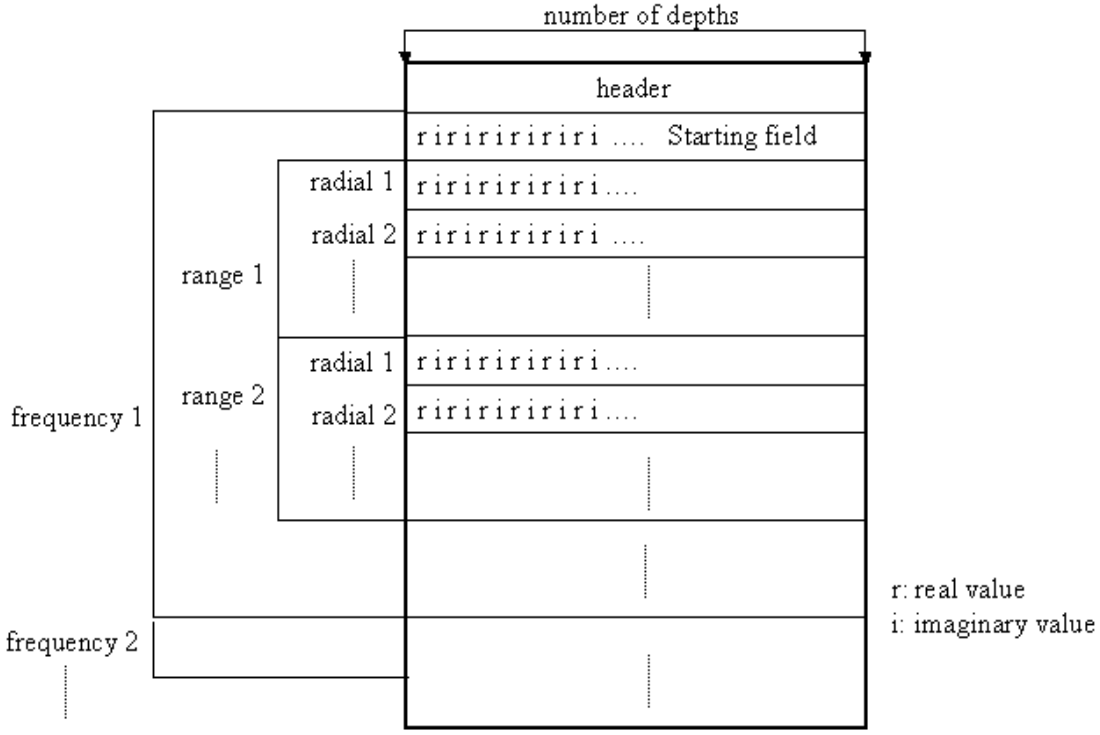

Figure 8. MMPE Output File Structure [Yonghoon 2000].

The binary output file must be post processed using two Matlab programs, called "peout1" and "peout2." The former reads the output file and the latter iteratively processes the uploaded information. In order to run, peout1 prompts for the name of the model output file and loads the file's information into the Matlab workspace. Then peout2 presents a list of options for calculation, as shown in Figure 9. The program can be run multiple times to obtain any additional information. From the six possible modes, the second option was selected. It computes the propagation loss for the whole target area in one run. This operation took an average of 5 minutes for each of the 77 data sets.

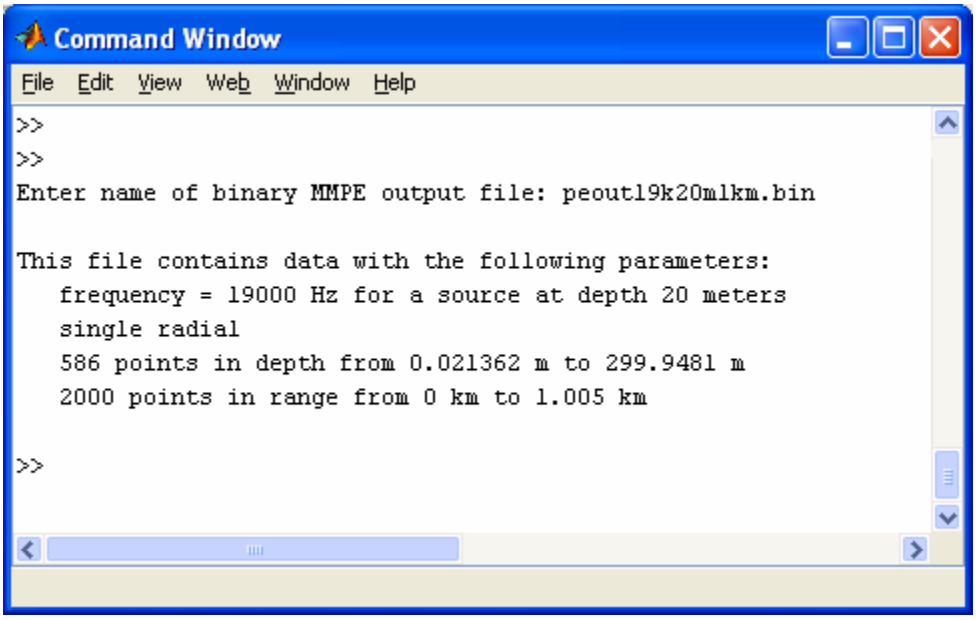

Figure 9. Screen Shots from Running peout1 and peout2.

The output of the peout2 program, for the option chosen, is a graphical representation of the propagation loss of the area showed in Figure 10. There is an option under File menu to save the propagation loss values into a Matlab data file.

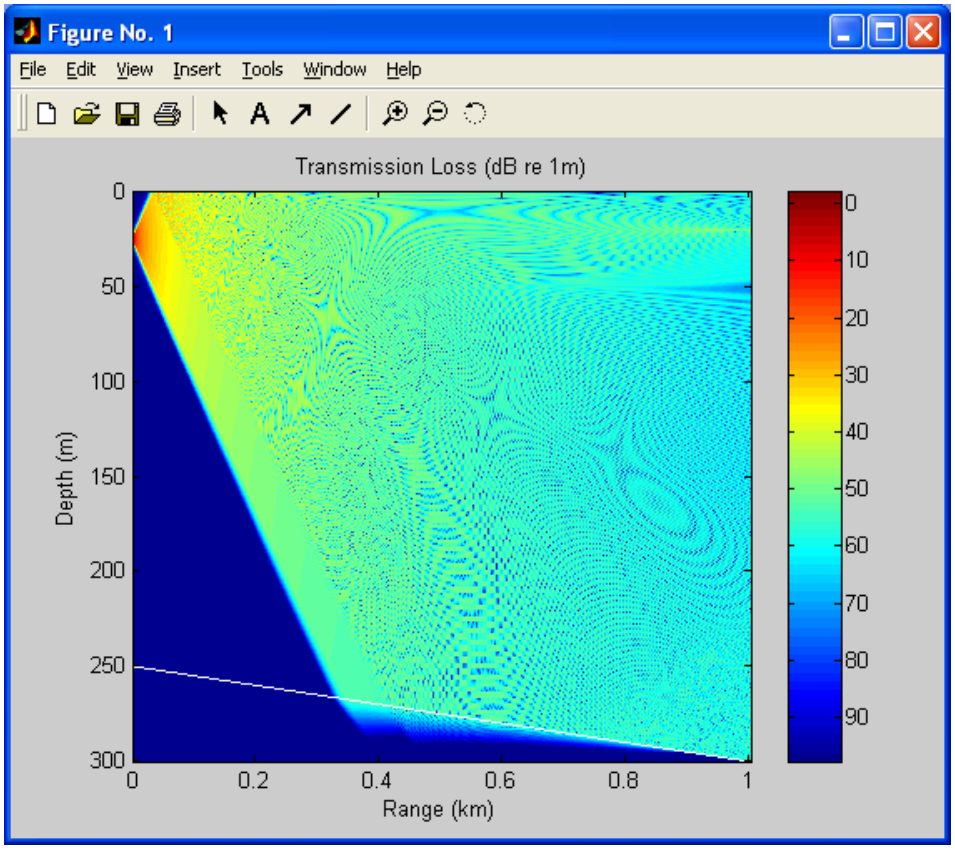

Figure 10. peout2 Output (Frequency = 19 kHz, Source Depth = 35m).

3. Data Format

The peout2 program produces a Matlab data file that contains a series of arrays with elements of the shortG type (five digits in fixed or floating point) as shown in Figure 11. The arrays of interest for our purpose are "tlpress," "rng," and "dep." The first contains the transmission or propagation loss of all the points in the range–by-depth slice. This set represents a 586 x 2000 matrix. The second holds the range points where the measurements were made (a 1x2000 matrix). The third, a 1x586 array, holds the depth of the calculated data points. In order to analyze the data, we had to match the three arrays so that an element of the "tlpress" array is tagged with the correspondent "rang" and "dep" elements.

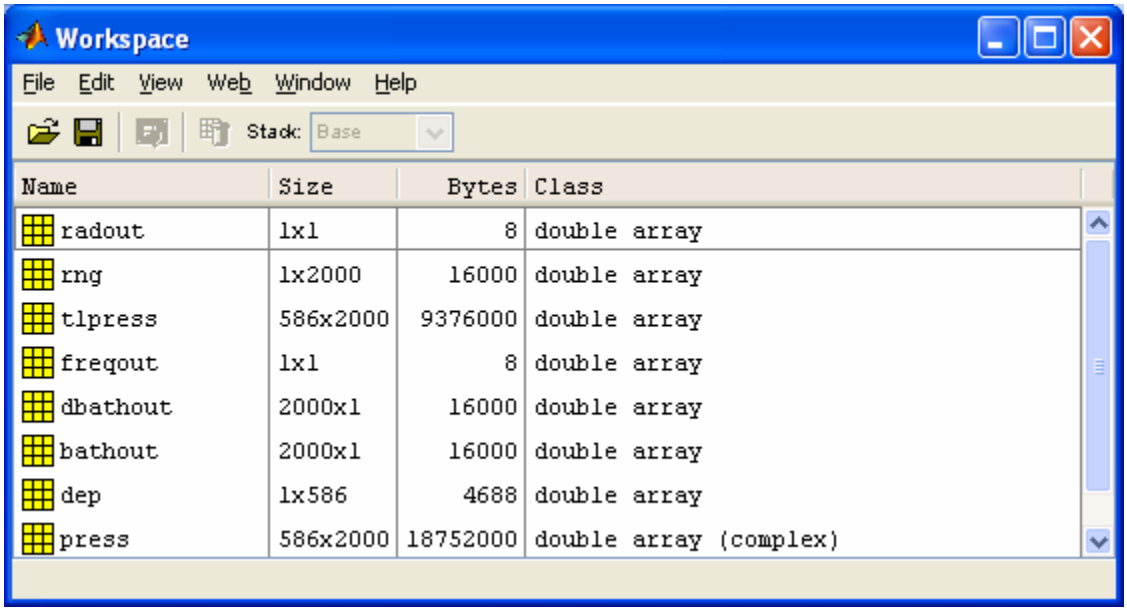

Figure 11. peout2 Output Array Data Structure.

C. INITIAL ANALYSIS

The data obtained by the Matlab program, peout2, was first structured in an Excel™ spreadsheet with the range as a title for the first column and increasing depth values in the subsequent columns. Excel only supports 256 columns and 64 rows; and the statistical analysis tools of Excel are intended to use with data arranged in rows. For our model we required, for each receiver depth, one row for each source depth, frequency, range, receiver depth, and propagation loss per row; leading to 2000 rows per receiver depth. Further, we have 586 different depth layers and eleven frequencies per source depth. Thus, the data cannot be analyzed in one set, nor can the data points be reduced as we observe great variations with small changes in range, and great propagation loss at very low ranges, as show in Figure 12. This is consistent with variations of 10 to 20 dB from a source at 1 kHz and three quarters of a mile in range, attributable to a three feet variation in depth due to tide variations. On other experiments, over a few kiloyards, variations of 5 dB were encountered [Urick 1983]. The propagation loss that we found with the model at shorter ranges is due to the shadow zones [Lysanov 1982] that do not receive sound rays due to the directionality of the sound emitter, as we can see on Figure 10. We also found up to 20 dB variations over half a meter (10 kHz 1 m depth). Note that if we do not vary the inputs the model will provide a deterministic output.

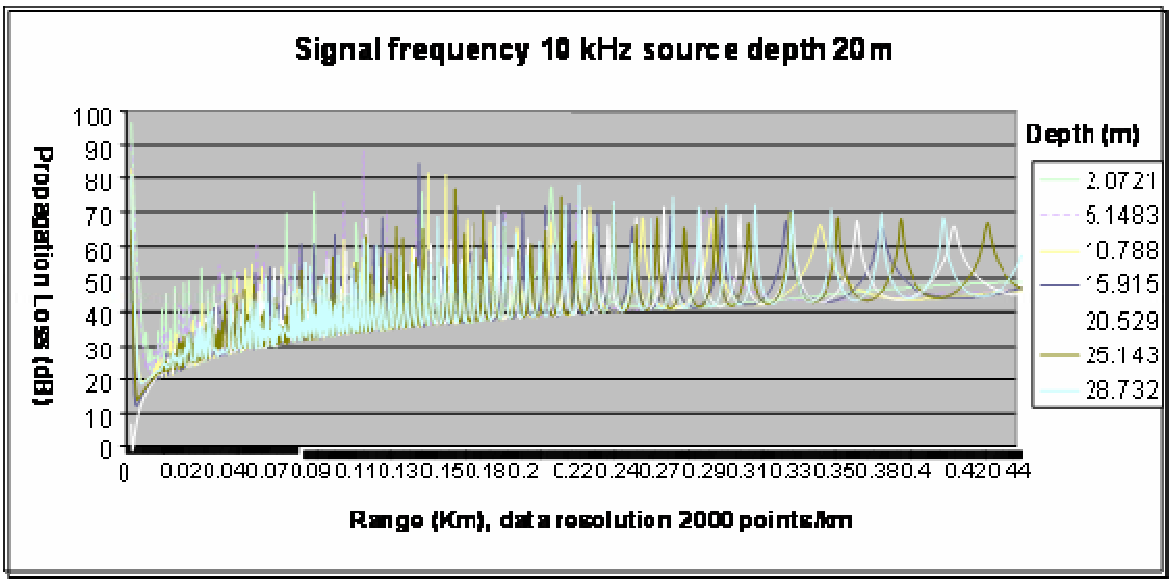

Figure 12. Propagation Loss vs. Range for Different Receiver Depths.

This deterministic nature of the model is due to the static definition of the environment. In an attempt to address this limitation, a modification of the MMPE, by its developers, with artificially induced random errors (0.1% on sound speed and 1% on the depth, bottom and sound speed gradients) was created but showed variability that was appreciable at ranges beyond 10 km [Smith 1999].

In order to have a more realistic model approximation, we need to have variability that resembles the reality of the sound propagation fluctuations due to random and cyclic ocean variations. In order to meet the time constraints that a network simulation requires, the model needs to be non iterative. A preliminary approach should look like this:

$$PL(t) = m(f, s, d_A, d_B) + w(t) + e()$$

where

$PL(t)$: propagation Loss while transmiting from A to B at time t

$m()$: is the propagation loss without random and periodic components; obtained from regression using data generated by MMPE model

f : frequency of transmitted acoustic signals in kHz

d_A : sender depth in meters (measured from sea surface)

d_B : receiver depth in meters (measured from sea surface)

r : horizontal distance between A and B (called range in MMPE model)

s : distance between A and B : $s = \sqrt{(d_A - d_B)^2 + r^2}$

$w(t)$: periodic function to approximate signal loss due to wave movement

$e()$: signal loss due to random noise or error

The functions $w(t)$ and $e()$ are not considered in the MMPE model and are introduced as an attempt to provide for better representation of the dynamic features of the ocean environment. The function $m()$ originates from the data that we collected and is arranged in data sets representing the 586 x 2000 data points plus the range, receiver depth, frequency and source depth markers. To perform a regression, we need to arrange this information in five columns: source depth, frequency, range, receiver depth, and propagation loss. The number of rows to accommodate the raw data is on the order of seventy-seven million. Thus, we cannot accommodate and analyze the data in one set nor a single program run. Our first approach was to separate the data into sets of one frequency and source depth and analyze them separately. However, we would need approximately one million rows just to do that and, as noted, Excel™ can not handle that high volume of data. We turned to S-Plus (version 7.0 for Windows™ build 7187) from Insightful Corporation to perform the regression analysis. In order to smooth and reduce the amount of data to a manageable level, we applied a smoothing technique that takes a 5 x 5 data point matrix and averages the 25 values, and then places the result in the center (third value in range and third in depth) as shown in Figure 13. This effectively will reduce irregularities that might prevent us from obtaining the correct approximation, while reducing the number of rows to a few million, which is manageable with S-plus 7.0.

$$\begin{pmatrix} a_{11} & \cdots & a_{15} \\ \vdots & \ddots & \vdots \\ a_{51} & \cdots & a_{55} \end{pmatrix}$$

$$a_{33} = (a_{11} + \ldots + a_{55}) / 25$$

Figure 13. Data Smoothing.

D. FORMULA DETERMINATION

1. Data Reduction

To smooth the data we created a Matlab program that reads five data points in range with five in depth, averages them, and assigns the value to the central data point and continues doing so through the lateral distance. It then repeats the process for the next five rows of data points in depth, from the beginning of the range, until the end of the data set is reached. After smoothing and reducing the data to a manageable level, we still needed to identify shadow zones and outliers in order to provide a regression fit as close to the "smoothed" model as possible. As can be observed in Figure 14. at closer ranges we have higher values of propagation loss that are realistic, due mainly to the shadow zone, as shown in Figure 10. We observed that the first 50 meters of the data are highly susceptible to such effects and we considered that at such short ranges, a properly oriented source will provide with a more realistic propagation loss and thus we decided not to consider data at closer ranges than approximately 50 meters.

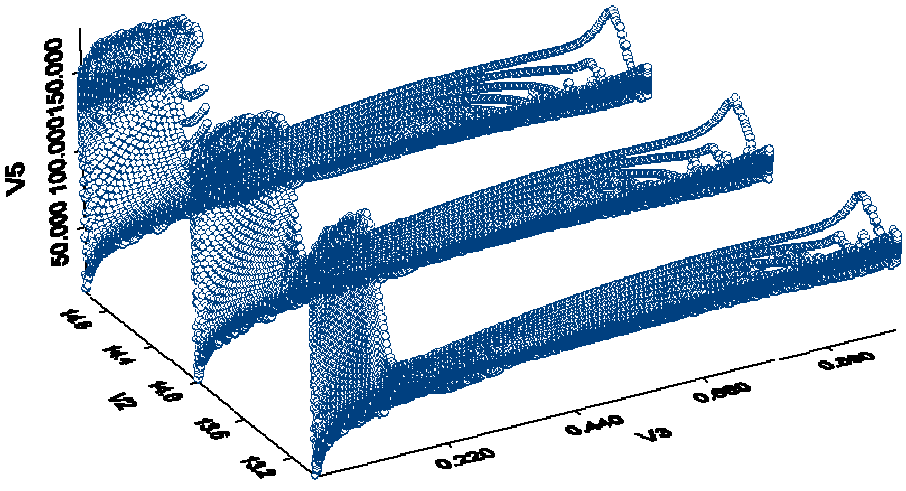

Figure 14. Smooth Data Outliers (V2: freq (kHz); V3 range (km); V5: dB loss).

For the purposes of this study, any propagation loss for data points below approximately one hundred meters will be not be considered. This has the benefit of reducing the amount of data to process. The final data set that is object of this study consists of approximately 1,100,000 data points and a sample of the data takes the form shown in Figure 15. It appears more uneven because of the smaller PL scale. In fact, the maximum PL value is about half of that in Figure 14.

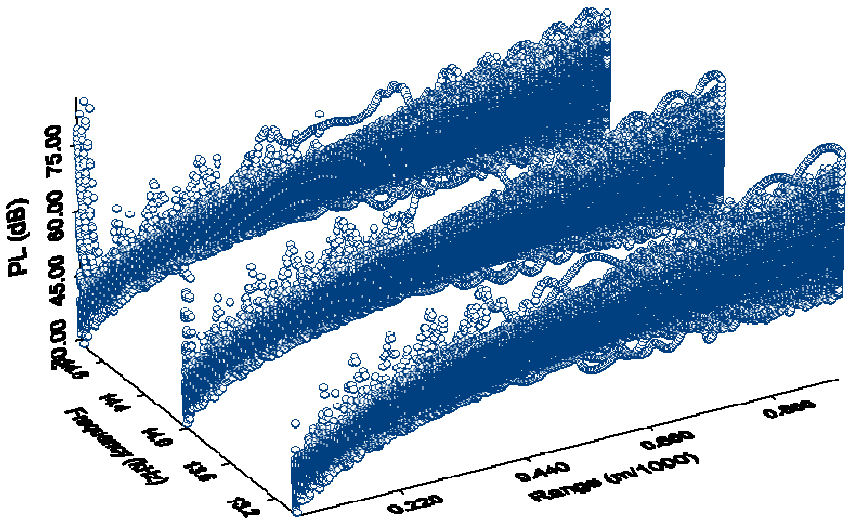

Figure 15. Smoothed Data (Complete Set).

2. Determining $m()$

Our approach was to build a model using regression techniques. According to our first data analysis and the logarithmic nature of the data, we believe a nonlinear regression is the best option to provide an approximation to the model. A nonlinear regression requires a preliminary model to calculate the coefficients [S-PLUS 2005]. The component determination of the approximation model is as follows.

The MMPE model in the post-processing phase applies the following formula as the last step in the calculation of the propagation loss [Yonghoon 2000]. This formula is similar to the Thorp equation.

$20\log_{10}|\psi(r,z,\varphi)| - 10\log_{10} r \ \ [dB\ re\ 1m]$

where:

ψ : is a function of the cylindrical coordinate system (r,z,φ)

$dB\ re\ 1m$: reference dB at 1 meter form source

Next, we explored the absorption coefficient at a depth of 3000 feet and at 4° Celsius [Urick 1983].

$$\alpha = \frac{0.11 f^2}{1+f^2} + \frac{44 f^2}{4100+f^2} + 2.75 \text{ x } 10^{-4} f^2 + .003 \quad [dB / Kyard]$$

We also introduced the following formula to account for variations related to the depth layer and the receiver-sender depth difference in the model.

$$a4 * d_B + a5 * \log(s) + a6 * s + a7 * \log(d_A) + a8 * \log((d_A - d_B)^2)$$

where $a4 - a8$ are constant coefficients to be determined, by regression analysis.

The data obtained and the model was incorporated into S-PLUS and a regression analysis performed with the options shown in Figure 16.

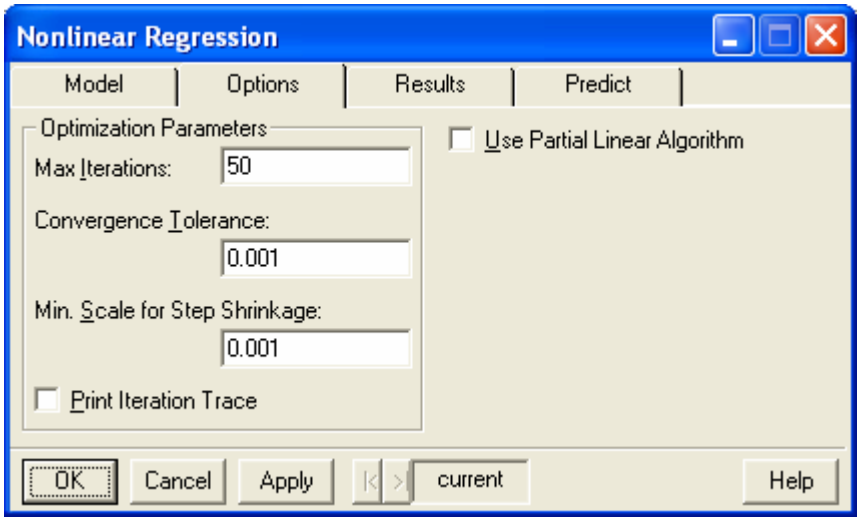

Figure 16. S-PLUS Nonlinear Regression Options.

The regression was executed on a computer with two Gigabytes of RAM and 2.4 GHz 64 bit processor. This is worth mentioning because S-PLUS consumes almost all the available memory, and subsequent regressions for different areas will require an equivalent amount of RAM to perform the processing.

After several iterations, the formula that we believe provides the best fit for the model is:

$$PL = \log\left(\frac{(s)^{a1}(d_A)^{a2}(d_A - d_B)^{a3}}{a5(s*d_B)^{a4}}\right) +$$
$$\left(f^2(a6/(1+f^2)+40*/(4100+f^2)+.000275)+.003\right)*(s/914)+$$
$$a7*d_B + a8*s$$

The regression analysis of the model produced a residual standard error of 2.76558 dB, on 1168531 degrees of freedom.

When analyzing the residuals, it can be seen from Figure 17. that they follow a Gaussian distribution, with the following summary statistics:

Minimum value : -1.316667e+001

1st Quartile.: -1.328983e+000

Mean: -4.595910e-009

Median: -2.672755e-001

3rd Quartile.: 1.013644e+000

Maximum 4.661067e+001

Total Number of data points: 1.168537e+006

Standard Deviation: 2.765339e+000

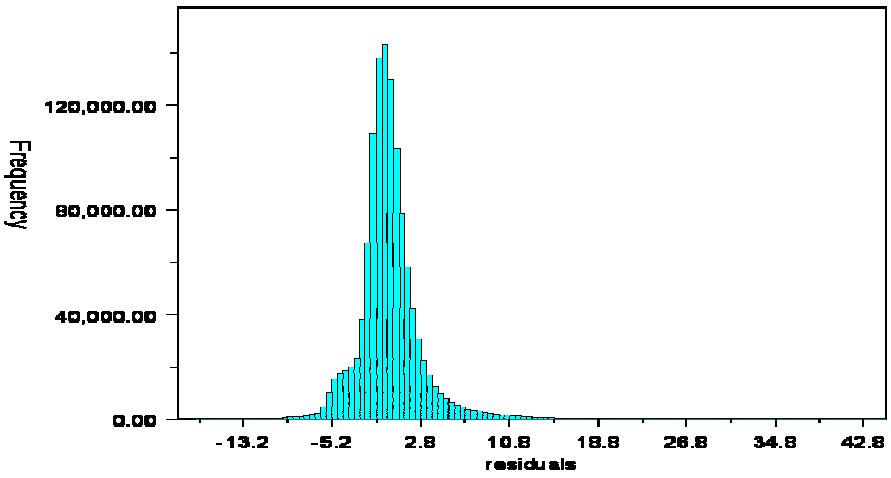

Figure 17. Residuals Histogram (dB).

The residual plot shows, at first impression in Figure 18. , that there are still points were there is not a good fit. A closer look at the data shows that the presence of such points is due mainly to the shadow zone and their can be mitigated by directionality of the sender furthermore in most UAN simulations the distance between any pair of nodes is typically much greater than 100 m.

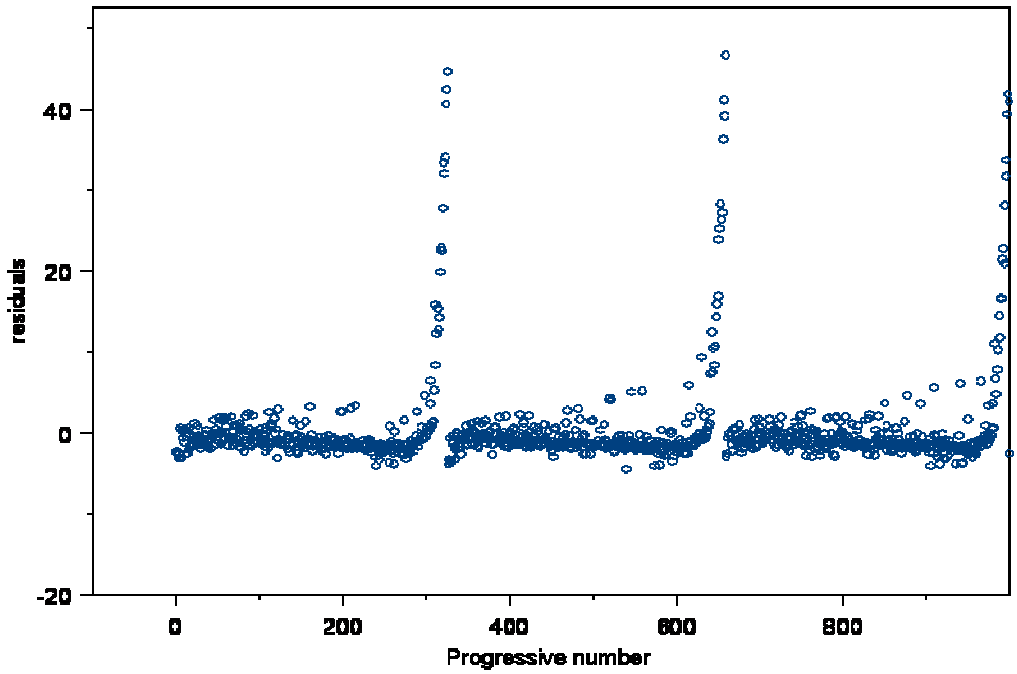

Figure 18. Residuals Plot (dB).

3. Determining $w(t)$

Our approach was to build a simple wave movement model based on the observation that under wave motion a water particle will oscillate around its location in a sinusoidal fashion [Dorn 1974]. That movement is represented as circular oscillations that reduce in radius as the depth of the particle increases. The length of that radius is dependent on the energy of the wave and is related to the wave height, as shown in Figure 19. Common waves are one hundred meters in wavelength and have an effect at depths up to 50 meters. [Dorn 1974].

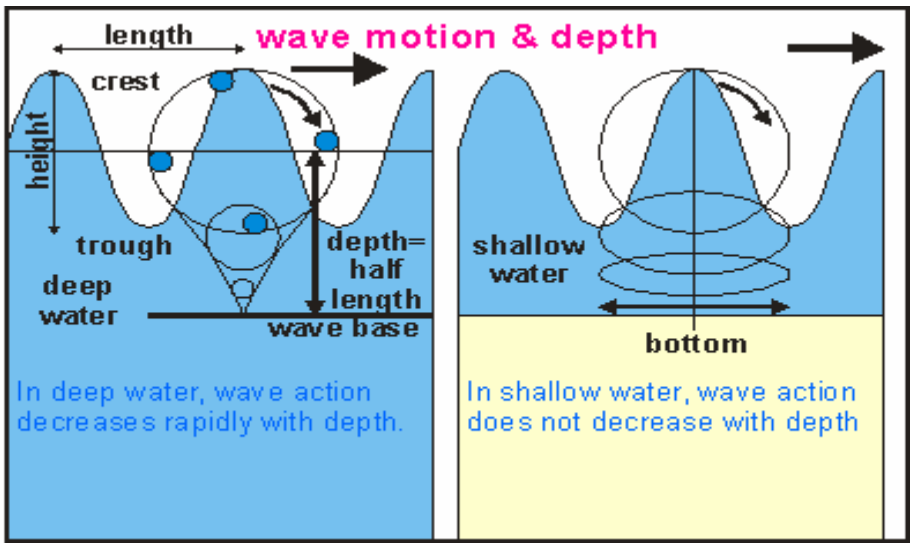

Figure 19. Particle Wave Movement [Dorn 1974].

To account for the change due to wave movement, a sample of the PL variations in 0.5 meter steps from 0 to 60 meters, this was made at 20 meters source depth, on the MMPE data (with no smoothing). The sample consisted of approximately 2.5 million points. The variability follows a normal distribution, as we can see on the histogram of Figure 20. with a standard deviation of 8.4851 and a mean of 0.081499 dB for a change in depth of 0.5 meter.

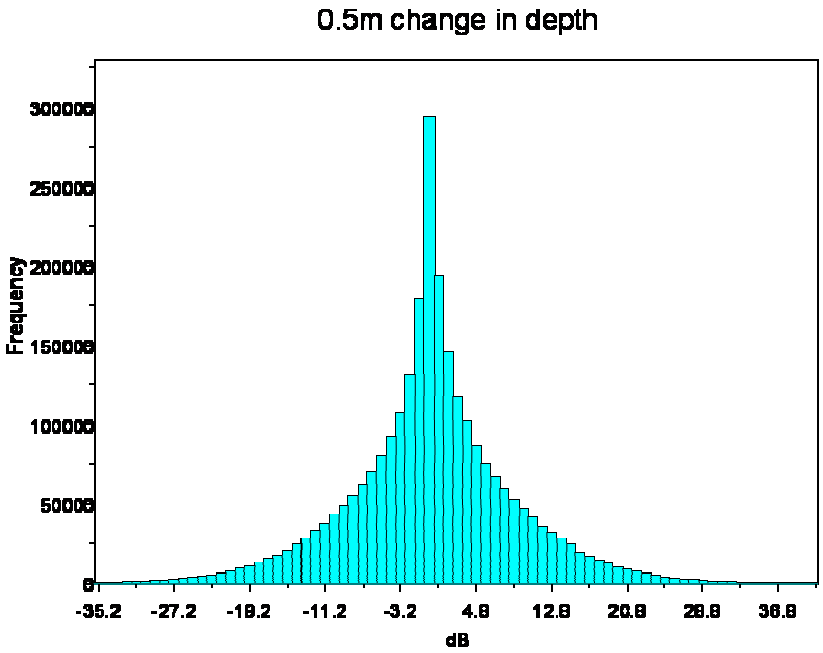

Figure 20. Histogram of Variation of PL Due to Change in Depth.

In order to model the wave effect can considerer the function:

$w(t) = h(l_w, d_B, t, h_w, T_w)E(t, T_w)$

where

$w(t)$: periodic function to approximate signal loss due to wave movement

$h()$: scale factor function

l_w: ocean wave lenght (m)

h_w:wave height (m)

T_w: wave period (seconds)

$E()$: wave effect function

This function contains the elements that resemble the node movement first by calculating the scale factor $h(l_w, d_B, t, h_w, T_w)$ and then the effect of the wave at a particular phase of the wave motion. The scale factor function is calculated as follows:

$$h(l_w, d_B, t, h_w, T_w) = \frac{(h_w * (1 - (2dB / l_w)}{0.5} abs\{\sin(2\pi(t \bmod T_w) / T_w)\}$$

The first term is divided by 0.5 because the distribution is based in a 0.5 meters vertical change. This is to increase the wave effect if the vertical change imposed by the wave movement to the node is greater than 0.5 meters. The term effectively has no units.

With each run of the model, $E(t,T_w)$ computes eight unique Gaussian random numbers for each potential receiver node. The random numbers are drawn from the estimated distribution of the wave effect at 0.5 meters.

$$E(t,T_w)=\left\{\begin{array}{ll} X_1 & (0)\le (t \bmod T_w)/T_w) \le (1/8) \\ X_2 & . \\ X_3 & . \\ X_4 & . \\ X_5 & . \\ X_6 & . \\ X_7 & . \\ X_8 & (7/8)< (t \bmod T_w)/T_w) \le (1) \end{array}\right\}$$

where

$\{X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8\} \sim Normal(\bar{X}, \sigma)$

$\bar{X} = 0.081$

$\sigma = 8.48$

It is important to compute a different set of random numbers for each node in order to model the fact the nodes will not be in the same phase of the wave motion at a given time. It should be noted that this wave effect model is somewhat simplistic. For example, the model does not consider the motion of the signal source. The use of an estimated distribution of PL variations at 0.5 meter depth increases is kind of ad hoc. The model is meant to be a first (rather small) step towards addressing a complex problem in the modeling of ocean acoustic propagation loss. A more accurate modeling of the wave effect is left for future work.

4. Determining *e*()

A random term was introduced to account for background noise. As the number of sound sources is large and undetermined, this random noise follows a Gaussian

distribution [Urick 1983] and is modeled to have a maximum of 20 dB at the furthest distance [Sozer 1999]. This is calculated by the following equation.

$$e() = 20(s / s_{\max})R_N$$

where

$e()$: random noise function

s: distance between sender and receiver

$s_{\max}$: maximum distance

R_N: random number, Gaussian distributed with 0 mean and variance of 1

E. CHAPTER CONCLUSIONS

There are many sound propagation models available and diverse statistical tools can be used to analyze those models. The process of obtaining a statistical approximation to those models in the pursuit of more efficient, hence more suitable, calculations for a network traffic simulation is the main focus in this chapter.

The MMPE model presents some challenges in the time it requires to obtain the data, as well as the amount of data produced, Even if we have access to a computer with two gigabytes of RAM we encounter out of memory errors that slow down the generation of the approximation. Basically, the solution that was developed will provide a reasonable approximation. The use of the wave movement, an attribute of the real environment, applied to the original data, will increase the fidelity of the propagation loss model. The random component will allow, to an extent, the model to account for background noise, also a characteristic of the acoustic environment in the ocean.

Next, we will present the implementation of the MMPE approximation model, with the randomness factors incorporated.

IV. IMPLEMENTATION

A. INTRODUCTION

This chapter first introduces the requirements for the implementation of the type of propagation loss approximation algorithm described in the previous chapter. Then it discusses the design aspects for building a module that can be used in a platform independent fashion to provide propagation loss figures for an underwater simulation. The module implements a time budget as a safety measure to recover from the occurrence of a problem with a given loss calculation. After that, the discussion shifts to a specific implementation within the OpNet environment, with details provided for the modifications needed, such as an update on collision detection, to adapt the new propagation loss model to an existing simulation.

B. REQUIREMENT ANALYSIS

We use the Unified Modeling Language (UML) to produce the requirement analysis and design diagrams, the modeling tool used was MagicDraw® version 9.5.

1. Introduction

a. Purpose of the System

In the rest of the chapter, we refer to an implementation of the propagation loss algorithm obtained in Chapter III as an Underwater Acoustic Network Propagation Loss Module (UANPL). The purpose of the UANPL is to provide an underwater acoustic network model with the means to make a more realistic physical layer approximation within the time constraints of a discrete event simulation.

b. Scope of the System

The UANPL will support the UAN simulation as an independent module that calculates and returns a propagation loss value for a given input parameter value set.

c. Objectives and Success Criteria of the Project

One of the critical success factors is the timeliness of the response of the module to the inputs with respect to the propagation time of the traffic that is "traveling" in the model. To satisfy these criteria, the objectives of UANPL module development are to:

- Provide a quick propagation loss calculation during the simulation;
- Compute a propagation loss calculation that considers the relative depths of the transmitter-receiver pair;
- Provide a solution reasonable enough to improve the fidelity of the existing model.

2. Proposed System

a. Overview

The system will carry out propagation loss calculation in an underwater acoustic network simulation, in a deterministic way.

b. Functional Requirements

- The model shall obtain a propagation loss in a deterministic way.
- The propagation loss model will provide the propagation loss at a specified range for given transmitter-receiver depth values;

c. Nonfunctional Requirements

(1) Usability.

- The system shall use a domain metaphor to represent the interactions
- The model shall provide an easy to understand interface;
- The system is intended for an advanced user.

(2) Reliability.

- The system will be operating in an intermittent cycle and shall be available 99% of the time the simulation is performed;
- In case of failure the system shall return an error message.

(3) Performance.

- The system shall perform transparently to the user;
- For the user, the overhead due to the propagation loss calculation for one transmission instance should be less than ¾ second.

(4) Supportability.

- The system shall be built in a modular way, following the loose-coupling and high-cohesion principles of software engineering, providing all documentation necessary to fully understand the different modules of the system, and facilitating ease of change to modules (algorithms);

(5) Implementation.

- The implementation shall be made in C.

(6) Interface.

- The system shall interface with other systems (modules) of a UAN simulation.

d. System Models

(1) Use Case Diagram.

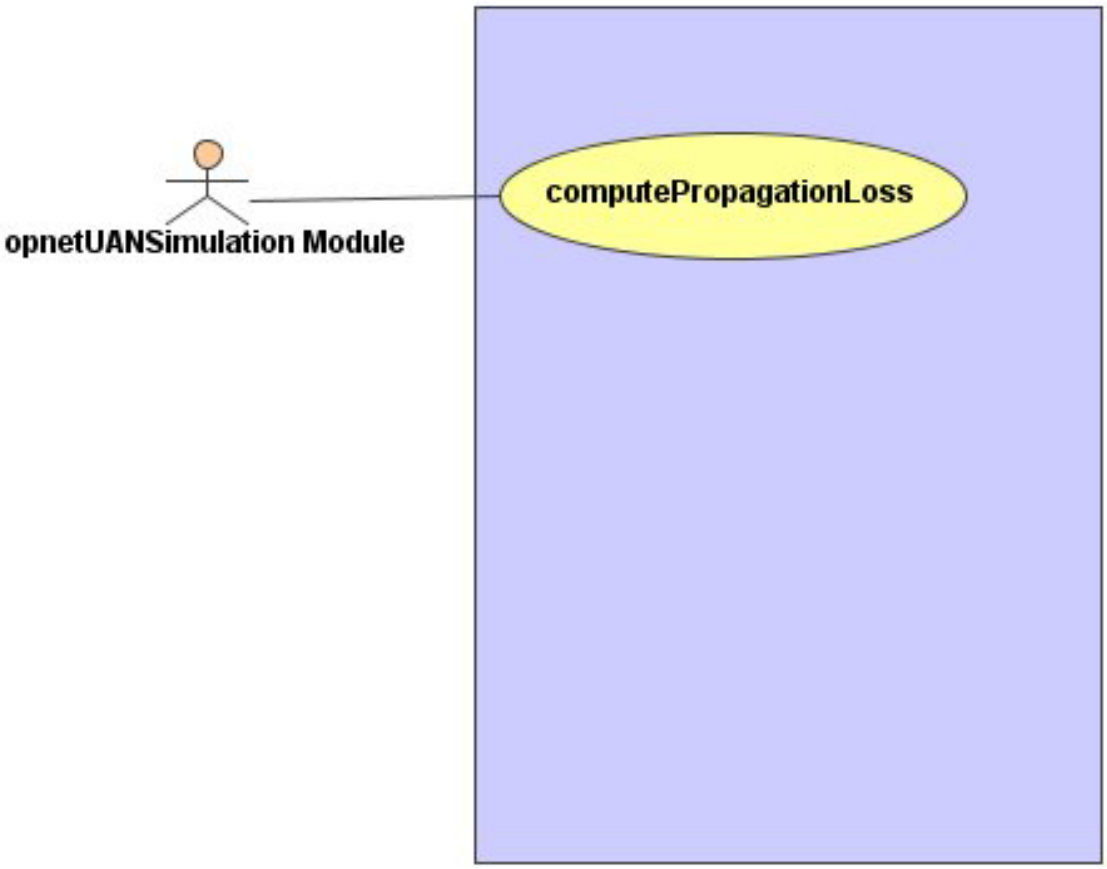

Figure 21. UAN Use Case at Top Level.

(2) Use Case Details.

- Use Case: UC01 – Compute Propagation Loss

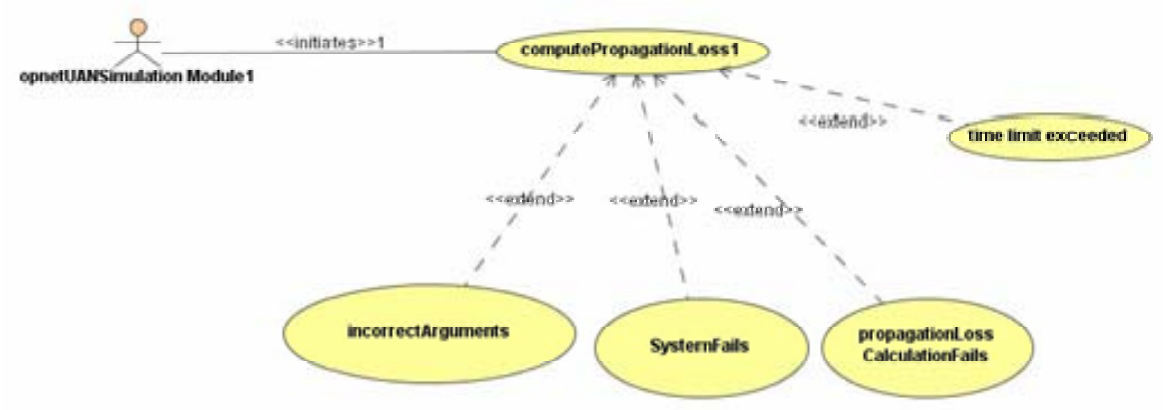

Figure 22. Use Case Exception Conditions.

Actors: OpNetUANSimulationModule (primary),

Stakeholders and interest:

OpNetUANSimulationModule – requires a timely propagation loss value;

Simulation Operator – wants to have a model that approximates reality;

Simulation designer – desires a module that produces an accurate propagation loss calculation in no more than ¾ of a second;

Entry conditions:

The system is running and ready to receive inputs; the UAN simulation is running and a packet is scheduled, and a propagation loss calculation is required.

Exit conditions:

The propagation loss value is returned to the UAN simulation module that requested the information;

Flow of events:

1. The UAN simulation requests a propagation loss calculation by providing the input such as depths of transmitter and receiver and distance between them to the propagation loss module.
2. UANPL accepts inputs.
3. UANPL processes the inputs and obtains the solution.
4. System returns the solution to the simulation module;

5. System is ready to receive inputs.

*a- When system fails

1- Return error message

*a- Time limit exceeded

1- Return error message

2a- incorrect Arguments to the Propagation Module

1- Return error message.

3a- Propagation Loss Calculation Fails

1- Return error message.

(2) System Sequence Diagrams.

- SSD01 – – Compute Propagation Loss

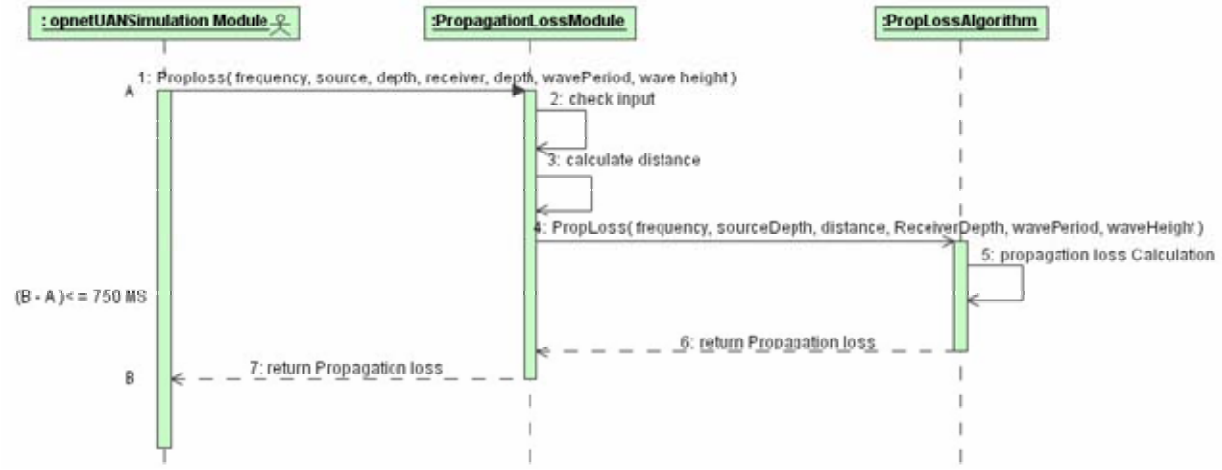

Figure 23. System Sequence Diagram SSD01.

(3) System Operation Contracts.

- C-01 – Compute Propagation Loss

Contract C-01 computePropLoss

Operation: computePropLoss(frequency:Int,sourcePoss:(Double, Double), depthT: Double, destPoss: (Double, Double), depth: Double)

Cross Reference: Use Case UC01 Compute Propagation Loss

Precondition: propagation Loss module is ready to receive inputs;

an instance p of PropagationLossAlgorithm exists;

an instance Pm of PropagationLossModule exists;

an instance O of OpNetModule exists;

an instance Tn of TranmiterNode exists;

an instance Rn of receiverNode exists;

Post condition: an instance pl of the propLoss association between Tn and Rn was created;

(4) Domain Model.

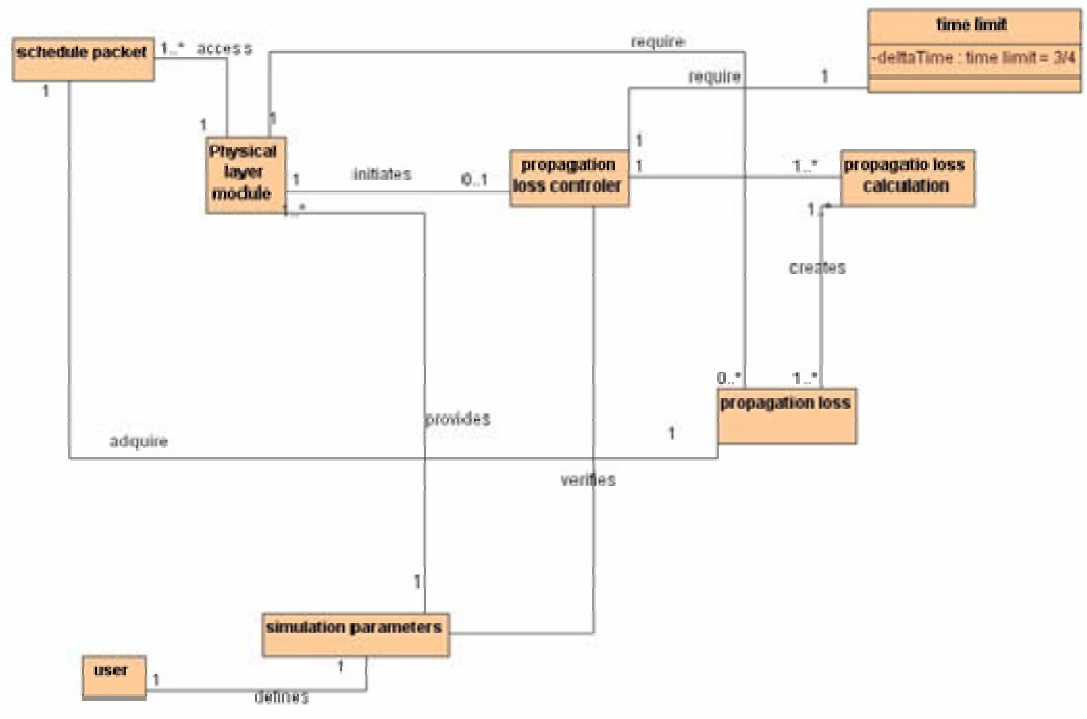

Figure 24. Domain Model.

C. DESIGN

1. Introduction

a. Purpose of the System

The purpose of the UANPL is to provide to the underwater acoustic network model with the means to make a more realistic physical layer representation.

b. Design Goals

The UANPL shall provide a propagation loss calculation that considers the water column depth, signal center frequency, depths of sender and receiver, absolute distance between sender and receiver (propagation distance), periodic wave movement, and a random acoustic noise factor. The resulting formulation, as described previously, is provided below.

$$PL(t) = m(f, s, d_A, d_B) + w(t) + e()$$

where

$PL(t)$: propagation Loss while transmiting from A to B at time t

$m()$: is the propagation loss with out random and periodic components

f : frequency of transmitted acoustic signals in kHz

d_A : sender depth in meters (measured from sea surface)

d_B : receiver depth in meters (measured from sea surface)

r : horizontal distance between A and B (called range in MMPE model)

s : distance between A and B : $s = \sqrt{(d_A - d_B)^2 + r^2}$

$w(t)$: periodic function to approximate signal loss due to wave movement

$e()$: signal loss due to random noise or error

The propagation loss calculation model must perform a calculation in at most 3/4 of a second. If the time constraint cannot be achieved then the model shall return an error.

The model shall be algorithm independent allowing modification to the implementation without changing the interface.

Once selected, the model shall be completely transparent to users. There is no direct interaction with users accessing resources.

c. Overview

The Software Design provides detailed technical data, system information, and other relevant information about the UANPL. This document includes a system architecture diagram, and for Use Case 01, as defined in the requirements phase, a design class diagram, state, and interaction diagrams.

2. Proposed System

a. Overview

The system shall be organized into a two-layer architecture, composed of an interface layer and an application logic layer. The purpose of this layering is to facilitate modular development. Each layer must meet rigorous interface specifications in order to interact with the other layer, making dependencies internal to a layer and allowing the incorporation of other propagation modules.

The interface layer should present to the OpNet physical layer module one stable interface and serve as a proxy for the propagation loss modules. It also functions as a watchdog to manage the time budget.

The application logic will be implemented by the propagation loss module. It is within this layer that the calculation logic is performed.

b. Architecture

(1) Subsystem Decomposition.

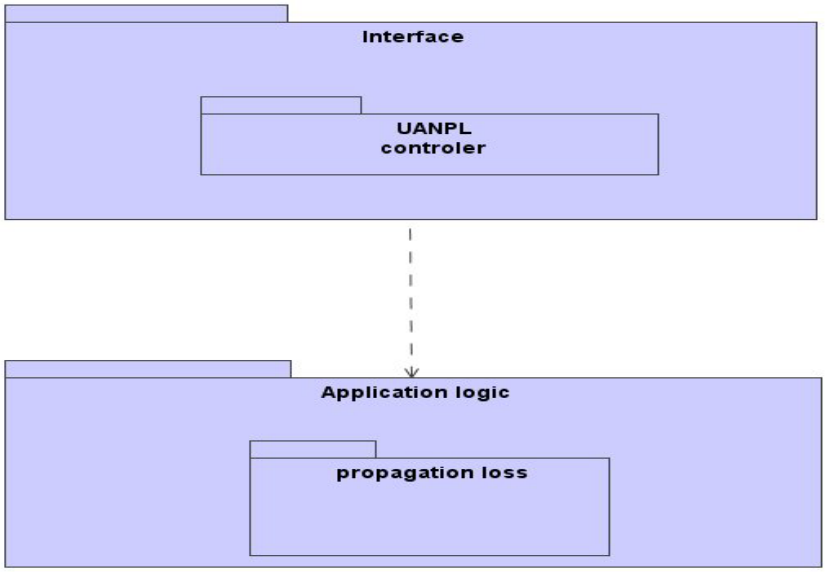

Figure 25. Subsystem Decomposition.

- Interface Layer

The interface layer is composed by the components that hold boundary objects. This subsystem allows the interaction of the OpNet physical layer module with the propagation loss module. This subsystem performs the argument checking and time constrains through the controller.

- Application Logic Layer

The application logic layer consists of the elements necessary to provide the system functionality. It contains the algorithm to calculate the propagation loss. This module can be changed to permit the implementation of different algorithms.

c. Global Software Control

The global control flow will be event-driven. The interface waits for a request from the OpNet physical layer module; performs parameter checking; and waits for the propagation loss, which it subsequently returns or, in the event of a time-out, returns an error condition. It then returns to the OpNet initial state.

3. Object/Class Description

The objects used in the application layer subsystem provide the interface and functionality to the system through their methods and attributes. The principal subsystems are the propagation loss method and the controller.

a. Controller

The controller object /Class, consists of the UanplController class. The main function is to determine if the arguments are correct, and, if so, passes those arguments to the propagation loss object /Class. It then starts and controls a timer.

Operations include:

Function	Arguments
propLoss	(frequency:INT, Source: LOCATION, depth:Double, receiver: LOCATION, depth:Double, waveP: Double, waveH: Double):Double
checkInput	(frequency:INT, Source: LOCATION, depth:Double, receiver: LOCATION, depth:Double, waveP: Double, waveH: Double): Boolean
calculateDistance	(Source:LOCATION, receiver: LOCATION) :Double
timer	(Status:Boolean) :Boolean

Table 1. Controller Operations.

(1) Operation Descriptions. propLoss (frequency:INT, Source:LOCATION, depth:Double, receiver: LOCATION, depth: Double, waveP:Double,waveH:Double):Double

The operation is used by the UanplController to receive input and return a propagation loss value or to indicate an error. In the case where a correct value is calculated within the time constraint the return is the propagation loss figure.

checkInput(frequency:INT, Source:LOCATION, depth:Double, receiver: LOCATION, depth:Double, waveP:Double:waveH:Double):Boolean

This operation is used by the UanplController to verify the compliance of the OpNet module with the interface contract and returns TRUE if the parameters are in the correct type and if the location frequency and depth are valid; FALSE otherwise.

calculateDistance(Source:LOCATION, receiver: LOCATION) :Double

This operation is used by the UanplController to obtain the distance (direct 3-dimensional distance) between nodes and then use that distance in posterior operations.

timer (Status:Boolean) :Boolean

This operation is used by the UanplController to abort the propagation loss calculation in the event of an expiration of the time budget.

b. Propagation Loss Module

The object PropagationLoss holds the object that performs the propagation loss calculation. This object receives the input from the Controller and performs the necessary computations to obtain the propagation loss figure.

Operations include:

Function	Arguments
propLoss	(frequency:INT, Sourcedepth:Double, Distance:Double receiverdepth:Double, waveP:Double,waveH:Double):Double
abort	(status : Boolean):Boolean

Table 2. Propagation Loss Module Operations.

(1) Operation Description. propLoss (frequency:INT, Sourcedepth:Double, Distance:Double receiverdepth:Double, waveP:Double, waveH: Double):Double

This operation is used by the propagationLoss object to retrieve the necessary information to perform the propagation loss calculation and returns the propagation loss as a type double.

abort(status : Boolean):Boolean

This operation is used to allow for a graceful algorithm termination in the case that the time budget expires.

4. Object/Class Diagram

In this iteration, we only developed the object class diagram in the scope of Use Case UC01 previously mentioned in the requirements phase.

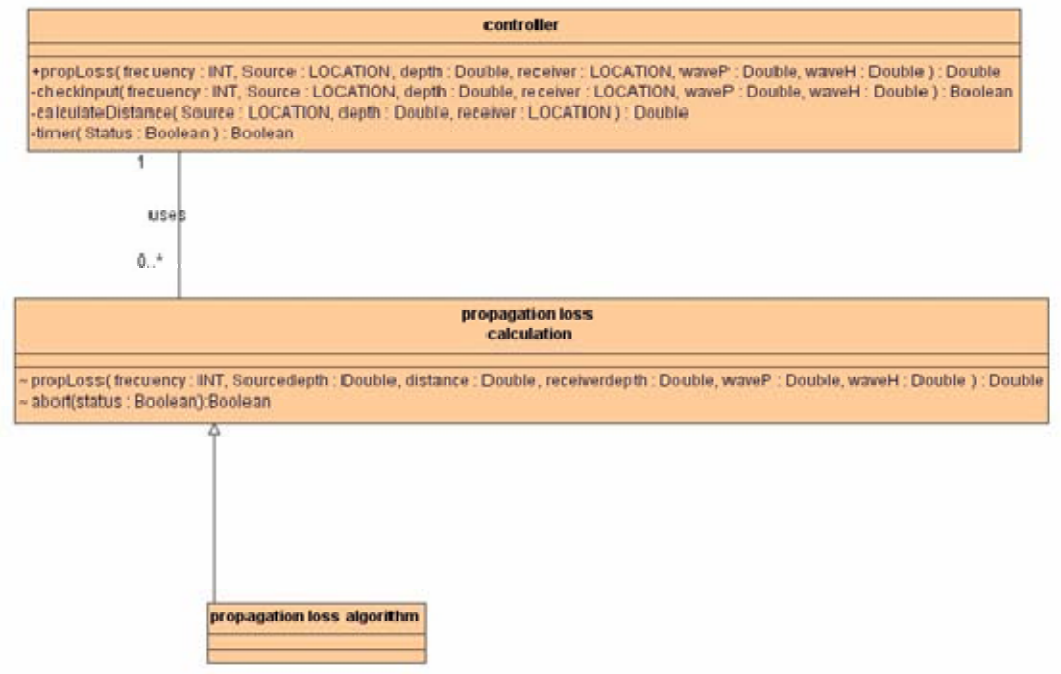

Figure 26. Object/Class Diagram.

5. System Sequence Diagrams

In this iteration, we only developed the system sequence diagram in the scope of Use Case UC01.

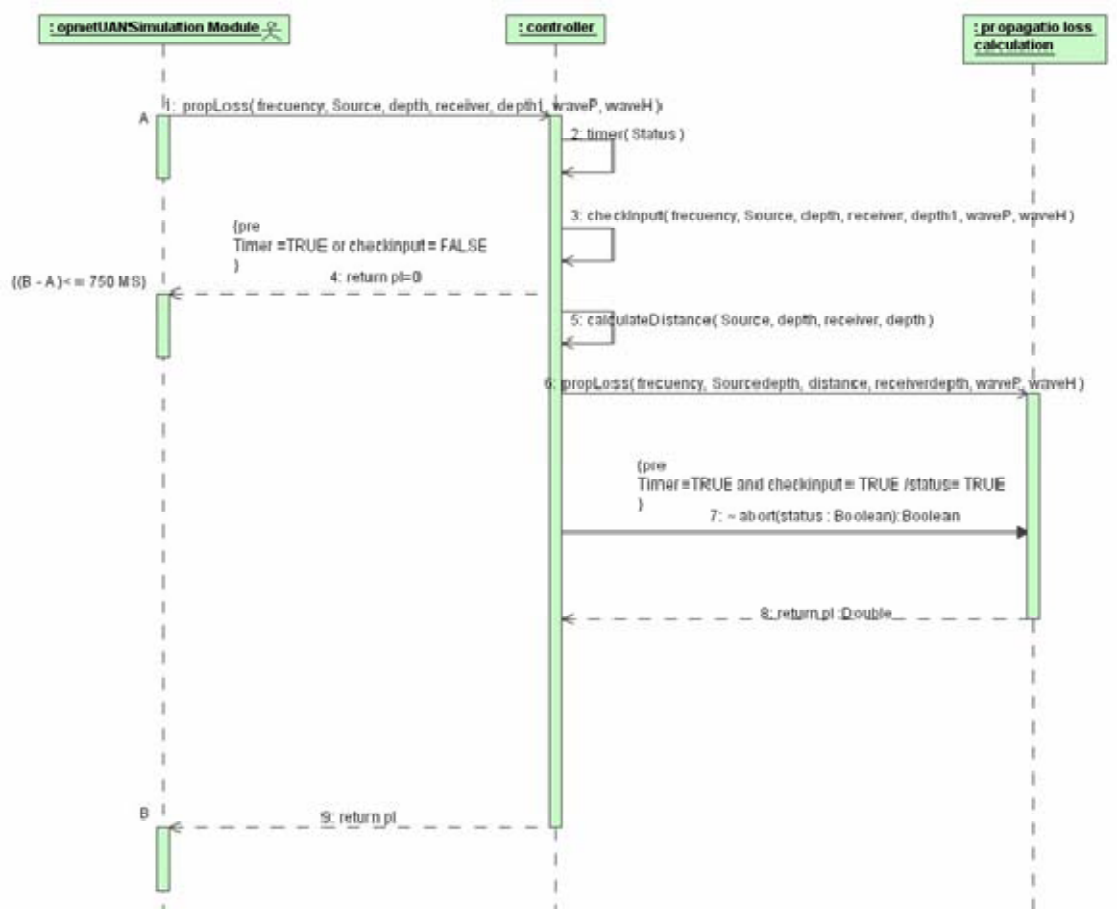

Figure 27. System Sequence Diagram SSD01.

6. State Diagrams

In this iteration, we only developed the system state diagram related to Case UC01.

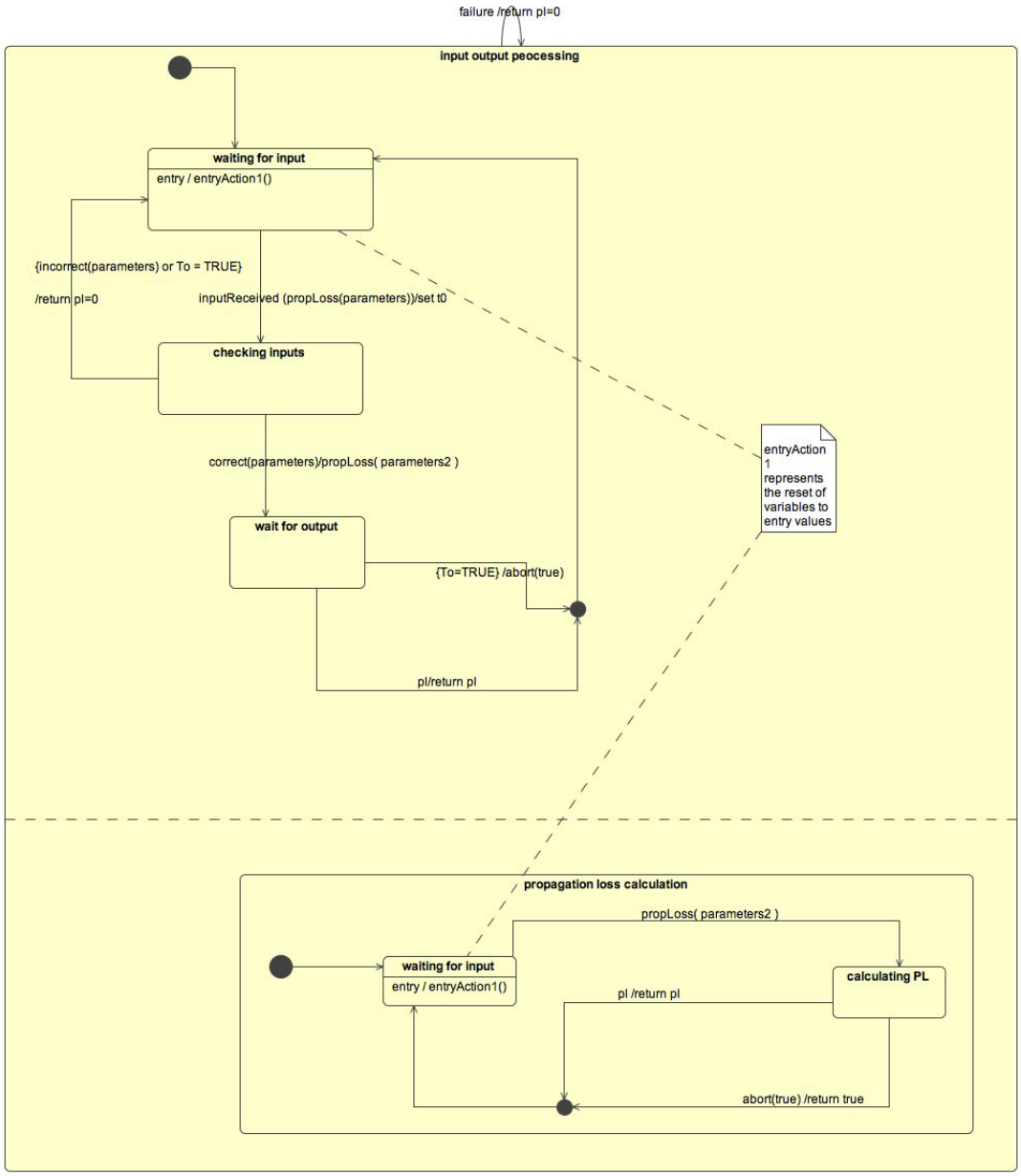

Figure 28. State Diagram.

7. Propagation Algorithm Module

The object PropagationLossAlgorithm holds the object that performs the propagation loss calculation. This object receives the input from the propagation loss module and performs the necessary computations to obtain the propagation loss. It does not perform time-bounds checking because the controller is responsible for that task.

Operations include:

Function	Arguments
getPloss	(frequency:INT, Sourcedepth:double, Distance:Double receiverdepth:Double, waveP:Double,waveH:Double):Double
plFor	(frequency:INT, SourceDepth:double, Distance:Double, receiverDepth: Double) : pLoss:Double
getTime	(void):t Time
getWavePL	(t:Time, waveP:Double, waveH:Double, receiverDepth:Double): plrDouble
addWavePL	(plr:Double,pLoss:Double): plW:Double
getRandomPL	(Distance:Double): plrDouble
addRandomPL	(plr:Double,plW:Double): plRw:Double
abort	(status : Boolean):Boolean

Table 3. Propagation Algorithm Module Operations.

a. Operation Description

getPloss (frequency:INT, Sourcedepth:Double, Distance:Double receiverdepth:Double, waveP:Double,waveH:Double):Double

This operation is used by the propagationLoss algorithm object to retrieve the necessary information to perform the propagation loss calculation and return the propagation loss, as a double.

plFor(frequency:INT,Distance:Double,SourceDepth:double, receiverDepth:Double) : pLoss:Double

This operation is used to obtain the propagation loss figure based on the regression model.

getTime(): t:Time

Obtains the current time of the simulation.

getWavePL(t:Time, waveP:Double, waveH:Double, receiverDepth :Double): wavePL:Double

This function calculates the effect of the wave movement related to the vertical node displacement.

addWavePL(wavePL:Double,pLoss:Double): plW:Double

This function incorporates the wave component to the propagation loss calculation.

getRandomPL(Distance:Double): plrDouble

This function calculates a random value that varies according to the range, returning the maximum value of 20 dB at the greatest distance.

addRandomPL(plr:Double,plW:Double): plRw:Double

This operation adds the random component to the propagation loss figure, wave effect included.

abort(status : Boolean):Boolean

This operation is used to allow for a graceful algorithm termination in case the time budget is exceeded.

b. Object/Implementation Diagram

For illustrative purposes, we only developed the object class diagram in the scope of Use Case UC1

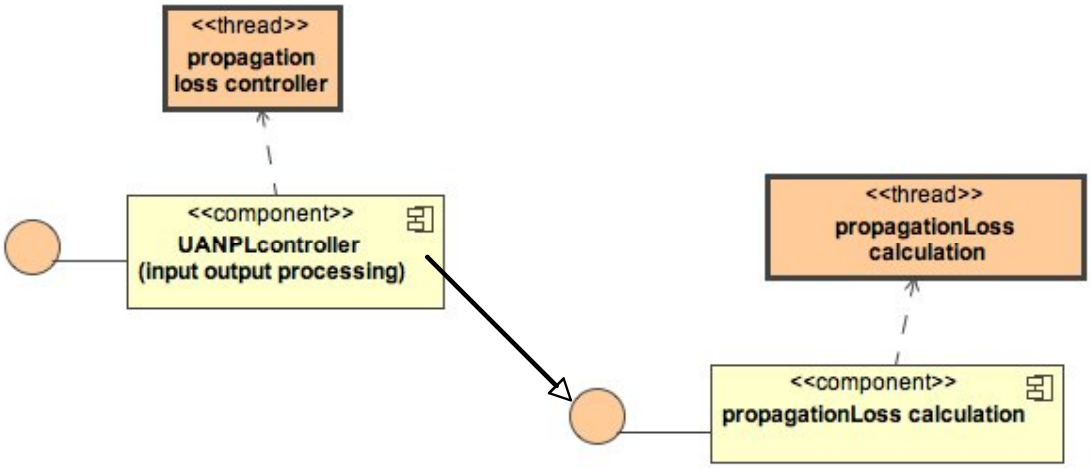

Figure 29. Implementation Diagram.

c. System Sequence Diagrams

In this iteration, we only developed the system sequence diagram in the scope of Use Case UC01.

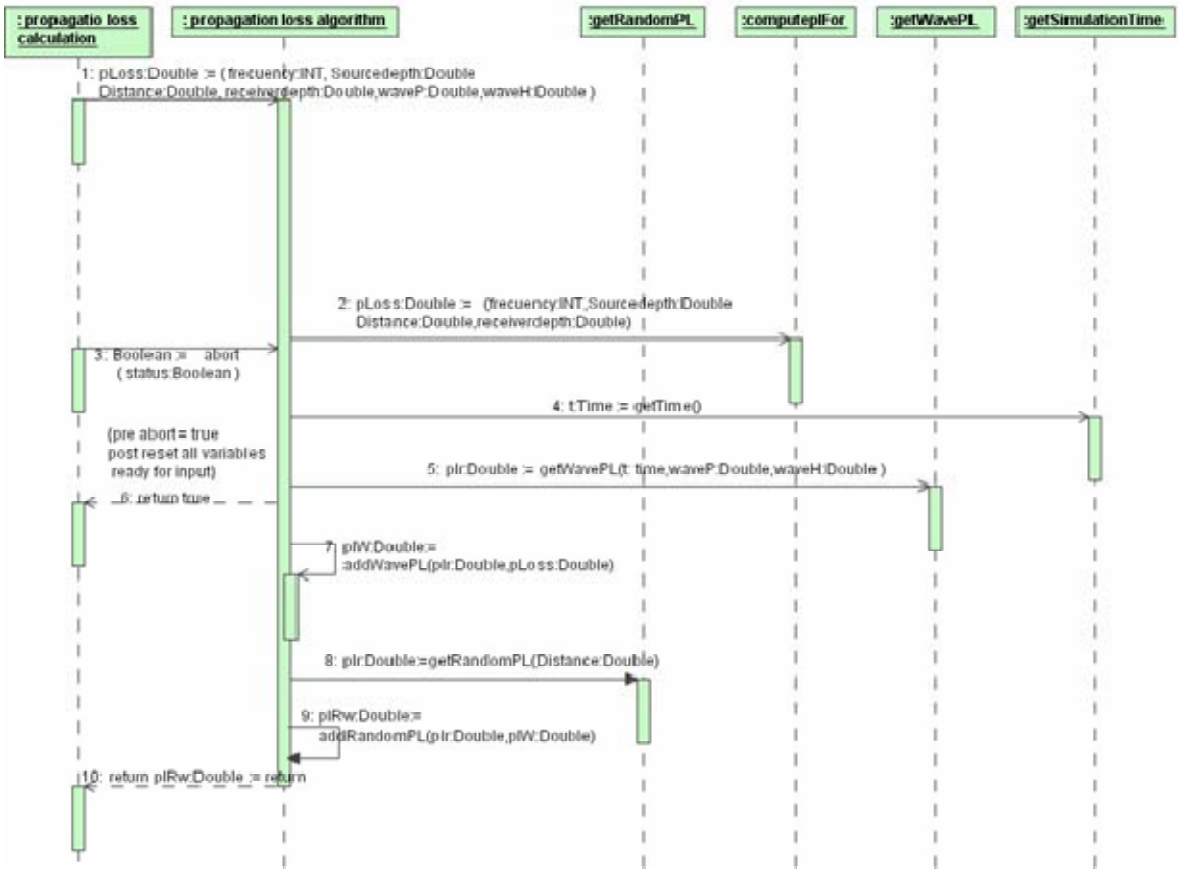

Figure 30. System Sequence Diagram.

d. Activity Diagrams

In this iteration, we only developed the system state diagram related to Case UC01

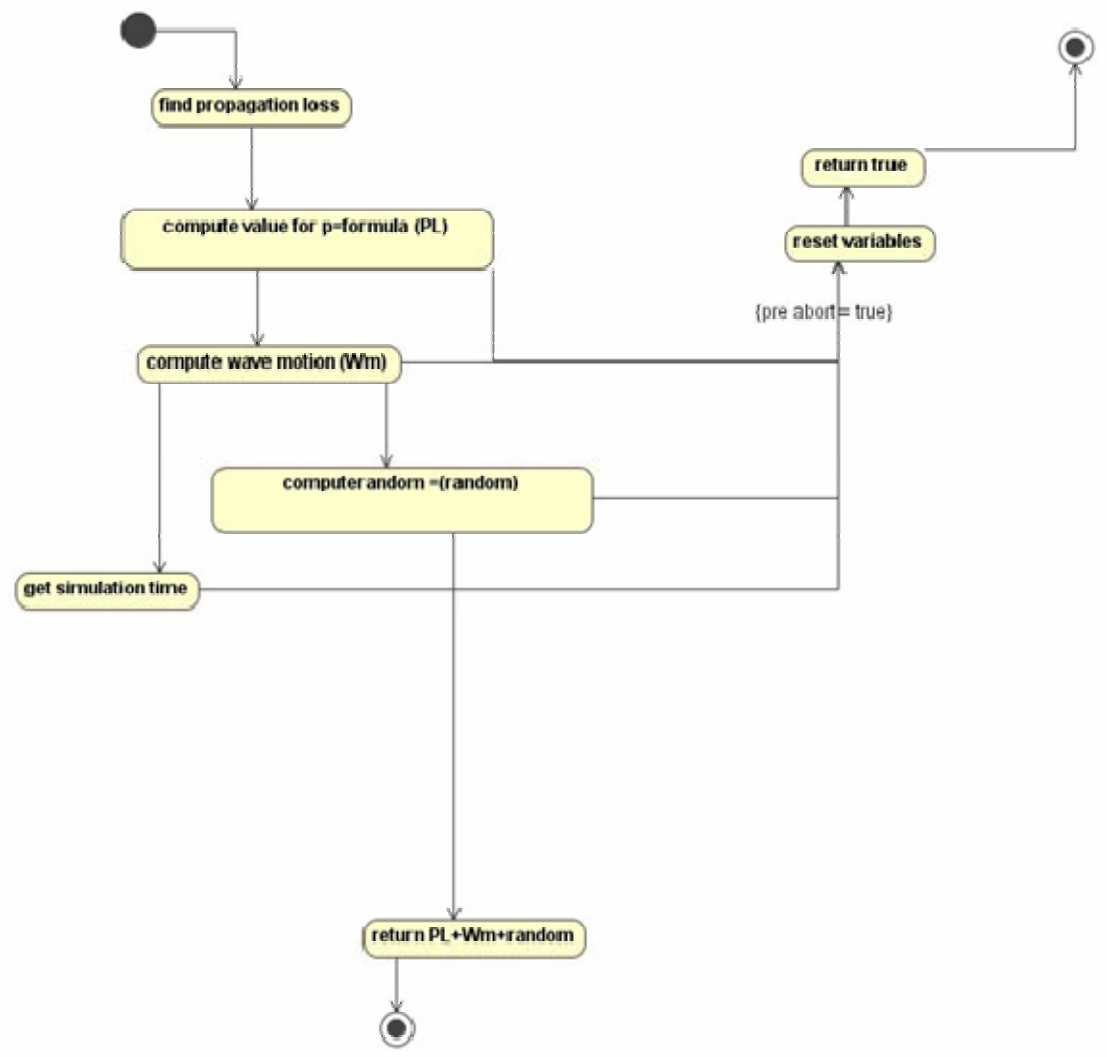

Figure 31. Activity Diagram.

8. Requirements Trace

Table 4 shows the dependency of the identified set of requirements on the object class (attributes and operations). This table only covers the design in the scope defined by Use Case UC01 in the requirements section.

Set of Requirements	System Object Classes	
	Controller	Propagation loss calculation
Time budget	Timer	Abort
Propagation calculation	propLoss calculateDistance	propLoss
Correct input	checkInput	

Table 4. Requirements Trace Matrix.

D. MODEL INCORPORATION

1. UAN Simulation

The UAN simulation selected to incorporate the propagation loss model is the one developed by Coelho [Coelho 2005]. As discussed in Chapter II, this simulation is implemented in OpNet. Its physical layer does not consider propagation loss factors other than a maximum propagation distance, which is set to 1500 meters.

The availability of the source code and the familiarity that we have with the model makes it a good candidate for incorporation of the new propagation loss model. We focus on the physical layer only and the modifications that are needed to incorporate the new propagation loss model into Coelho's work.

a. Model Physical Layer

The physical layer model created by Coelho determines the nodes that are within the maximum transmission distance. The process model is composed of five states (Figure 32.). The ones that are of most interest is the "define dest" state that creates the in-range node list, the "send" state that performs the transmission of the frames to the reachable nodes, and the receive state that retrieves the frame and sends it to the Media Access Control (MAC) layer.

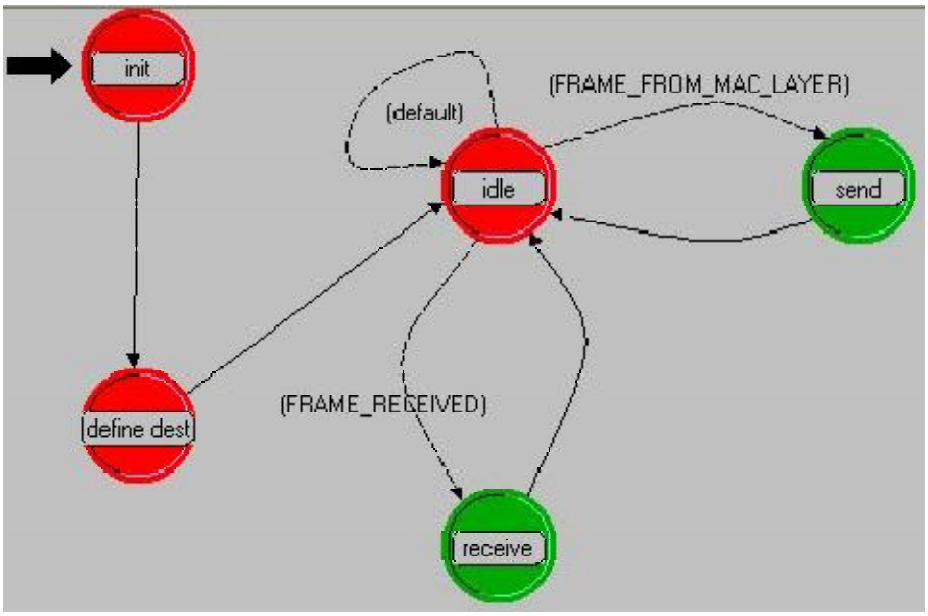

Figure 32. Physical Layer Process Model [Coelho 2005].

The state variables of this model need to be modified by adding additional ones to accommodate the parameters needed for the propagation loss model.

2. Physical Layer Modifications

The physical layer consists of the states showed in Figure 32. The corresponding OpNet module has six code block sections that are configurable by a developer to modify the behavior of the nodes. We made the following modifications to those sections:

a. Function Block

We added the following functions:

Function	Arguments
get Distance	(double myXpos, double myYpos,double myZpos, double otherXpos,double otherYpos,double otherZpos)
getPloss	(double freq,double senderDepth,double losDistance,double receiverDepth)
getWavePL	(double time, double period, double height, double receiverDepth)
addWavePL	(double wavePL, double pLoss)
getRandomPL	(double distance)
Disturbing	(Packet* frame)t

Table 5. Functions Physical Layer.

The purpose of the function, "disturbing," is to determine if a packet has been received with signal strength of such magnitude that it can "disturb" the ongoing reception of another previous packet, causing a collision to occur. In the actual implementation, this function is not used because it was emulated by not sending a packet to the neighboring node if the propagation loss is less or equal than the threshold. This effectively allows a collision to take place only if the packet has a propagation loss greater than the threshold and also increases or decreases the number of nodes a packet can reach. This function is explicitly mentioned because is a good point where a more elaborated collision detection algorithm can be implemented, in subsequent work.

b. State Variables

In this section, we added the last (bottom) six variables as showed in Figure 33.

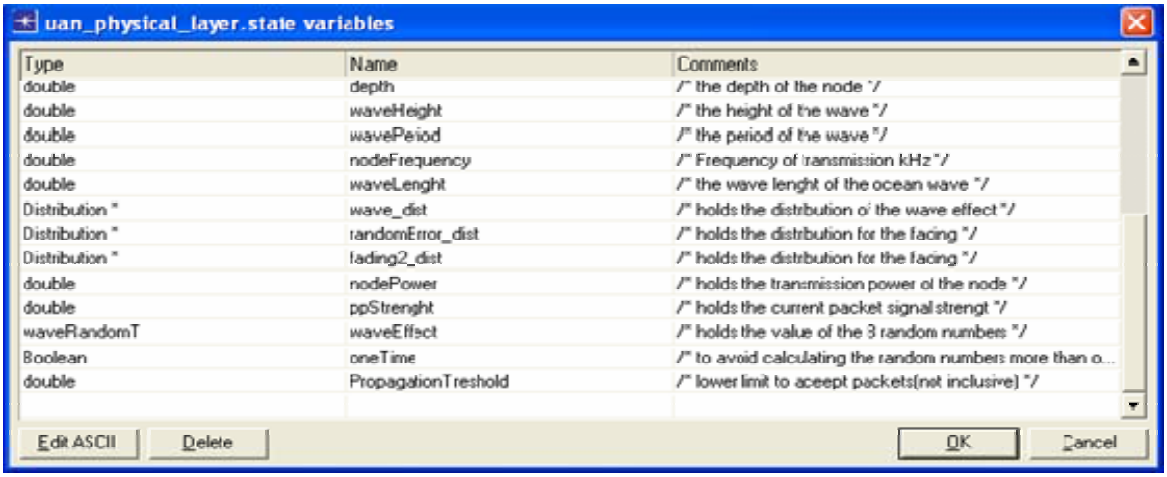

Figure 33. Physical Layer State Variables.

c. Temporary Variables

Here we added the following variables.

Type	Name
Packet*	copyFrameToo
UanT_Frame_FieldsP*	dummySignal

Table 6. Temporary Variables Physical Layer.

d. Global Attributes

In this section, we added the variables as showed in Figure 34.

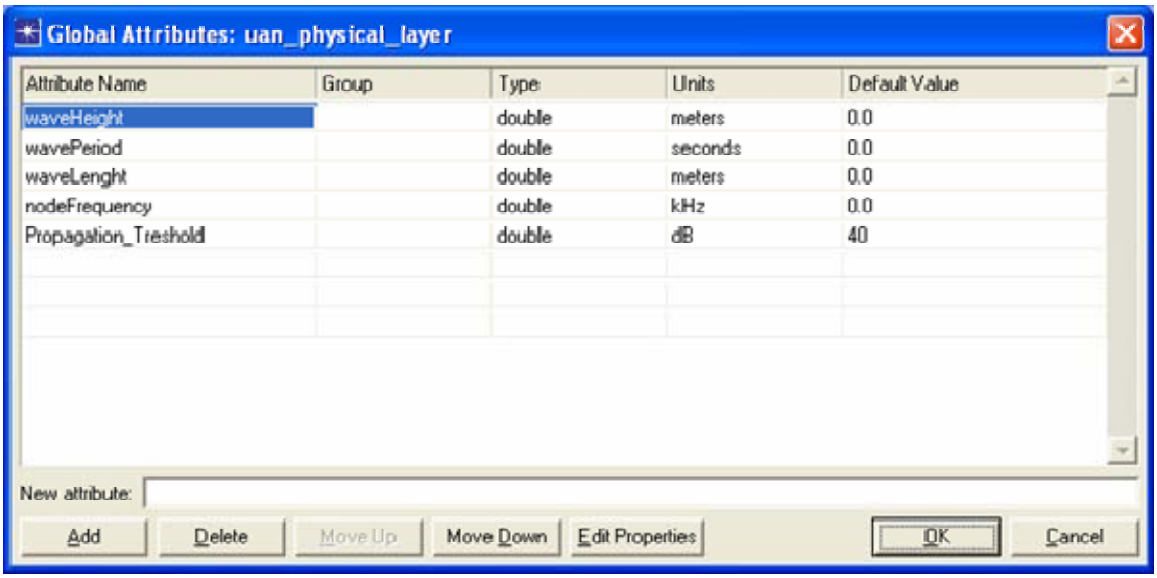

Figure 34. Physical Layer Global Attributes.

e. Receive State

To this state, we add the following logic to prevent a packet with a very weak signal from causing a collision.

...if (frame signal is strong enough)

send the frame to the MAC layer

else destroy the frame.

f. Send State

To this state, we added the following logic:

...open the frame that is going to be send to

the reachable node and add the correspondent

propagation loss

This is necessary because of the way the simulation is structured. The determination of the receiving nodes is done at this layer.

3. MAC Layer Modifications

The MAC layer modifications were restricted to the function block and to the state variables. The purpose of these modifications is to allow the model to deal with transmissions that have sufficient receive signal strength and to determine if a packet causes a collision.

a. Function Block

We added the following logic:

open the frame that has been received and obtain

the strength of the signal

b. State Variables

To this section, we added the last (bottom) variable as showed in Figure 35.

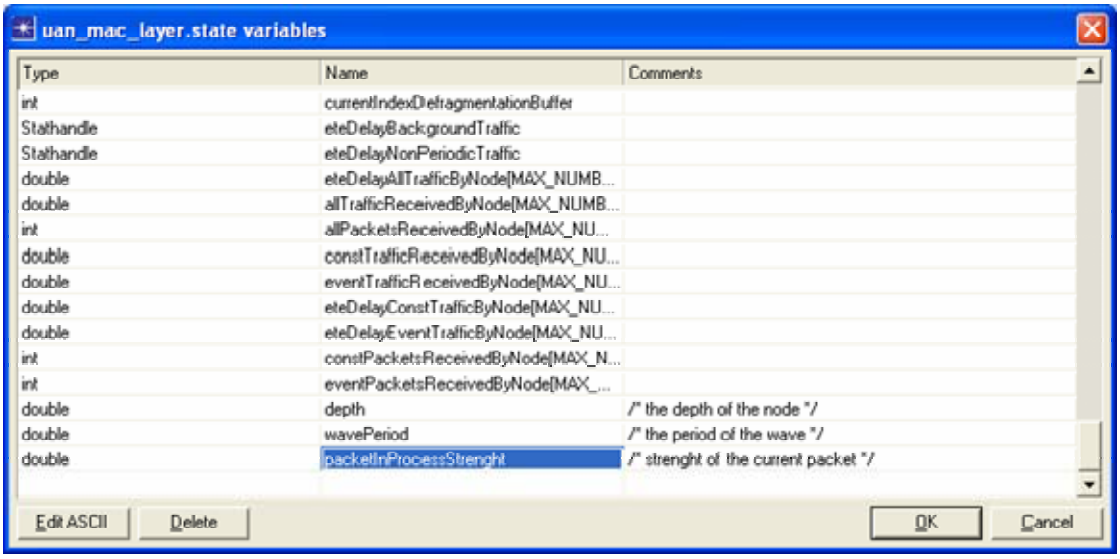

Figure 35. MAC Layer State Variables.

c. Header Block

Here we modified the UanT_Frame_Fields struct to accommodate a variable that holds the signal strength of the received packet. This permits the packet to provide its propagation loss value to the destination.

E. ADDITIONAL INFORMATION

The modifications to the physical layer C code can be reviewed in more detail in Appendix A where a part of the OpNet physical layer code (only the parts where the modifications where made) is presented.

F. CHAPTER CONCLUSIONS

The development of the requirements and a design specification prior to the actual implementation facilitates the development of a modular propagation loss module. However, the incorporation of the propagation loss module into an existing UAN simulation presents the challenge that the logic embedded in the simulation modules is not always suitable to handling the propagation loss impact on the receiving frames. Modifications that are not modular had to be implemented. A function to determine if the signal strength of co-mingling packets is sufficient to produce a collision was added to the model. Since the OpNet implementation was designed with interrupts, a receiving packet had to be intercepted before it is passed to the MAC layer to avoid errors.

The next chapter provides the results derived from executing the modified UAN model.

THIS PAGE INTENTIONALLY LEFT BLANK

V. SIMULATION

A. INTRODUCTION

This chapter will show the settings for running the simulation with the proposed algorithm. Also presented are the preliminary findings from execution of the resultant simulation. In order to make comparisons between the results of this model to the results of Coelho's model, the settings are similar to those used by Coelho. Furthermore, although the approximation algorithm only coarsely models dynamic sea motion, the behavior at the physical layer may even be more realistic than that determined by the MMPE model because of the inclusion of the wave effect and random errors.

B. SIMULATION EXPERIMENT

1. Network Topology

The MAC layer protocols that the Coelho model implements include contention-based message switching with collision avoidance (CA/MS) and an Aloha-like protocol with packet switching. One of the assumptions of the model is that the frames are transmitted without errors. The network is composed by three types of nodes: sensor nodes, relay nodes, and gateway nodes. The sensor nodes generate the network traffic and send traffic to the relay nodes. The relay nodes forward received frames to the next relay node until the packets reach the gateway, which will then retransmit the packets to another kind of media, such as radio. In all simulations run for this thesis, the packets are simply destroyed after they reach the gateway after statistics are collected.

Figure 36. illustrates the data flow towards the gateway in Coelho's original simulation setup. For this thesis, we limit the distance between a sensor and a relay to be around one kilometer, in order to be within the values of the data on which our regression was performed.

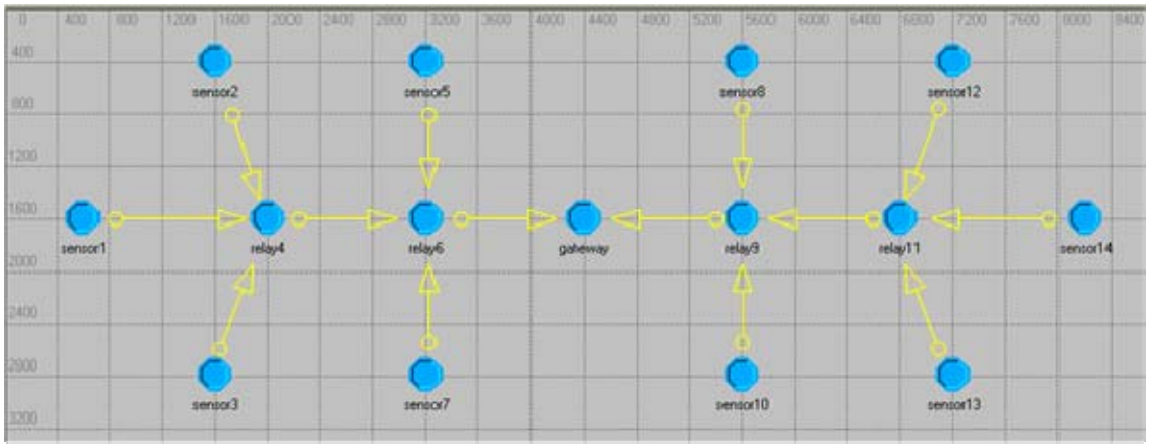

Figure 36. Tree Topology Network Layout [Coelho 2005].

Our simulation topology is very similar but with different transmission distances between nodes as showed in Figure 37. It also includes a depth parameter in meters, set as shown in Table 7.

NODE NUMBER	**DEPTH (meters)**
SENSOR ONE	20
SENSOR TWO	20
SENSOR THREE	25
RELAY FOUR	10
SENSOR FIVE	25
RELAY SIX	30
SENSOR SEVEN	15
SENSOR EIGHT	10
RELAY NINE	15
SENSOR TEN	10
RELAY ELEVEN	35
SENSOR TWELVE	12
SENSOR THIRTEEN	5
SENSOR FOURTEEN	14
GATEWAY	5

Table 7. Node Depth Setting.

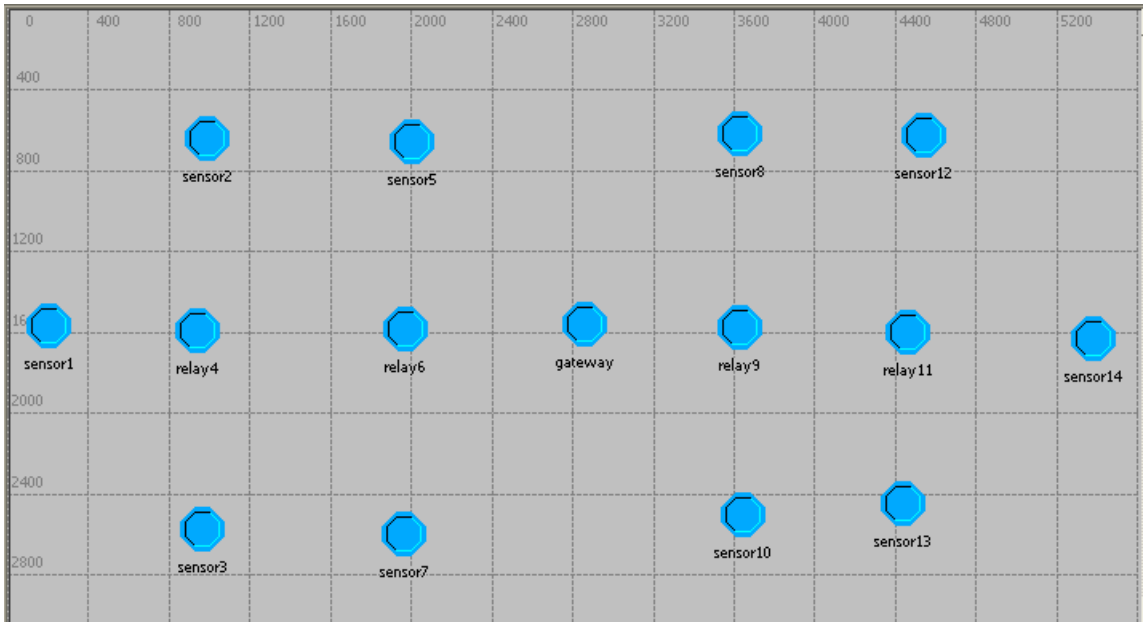

Figure 37. Tree Topology Network Layout.

2. Simulation Settings

The general settings of the simulation are defined as global attributes in OpNet. These attributes are selected at the beginning of each simulation run under the "configure run DES" (Discrete Event Simulation) window from the global attributes option in the input menu. The options that are subject to change and are of the interest to this work are shown in Figure 38. The units used are: nodeFrequency or the transmission frequency in kilohertz; waveHeight and wavelength in meters; and wavePeriod or the period of the dominant wave in seconds. The wave parameters are fixed in our experiments, with a wave height of four meters, a wave length of 100 meters, and a wave period of five seconds.

The Propagation_Threshold, the loss value beyond which received packets will be ignored, is input for each simulation run.

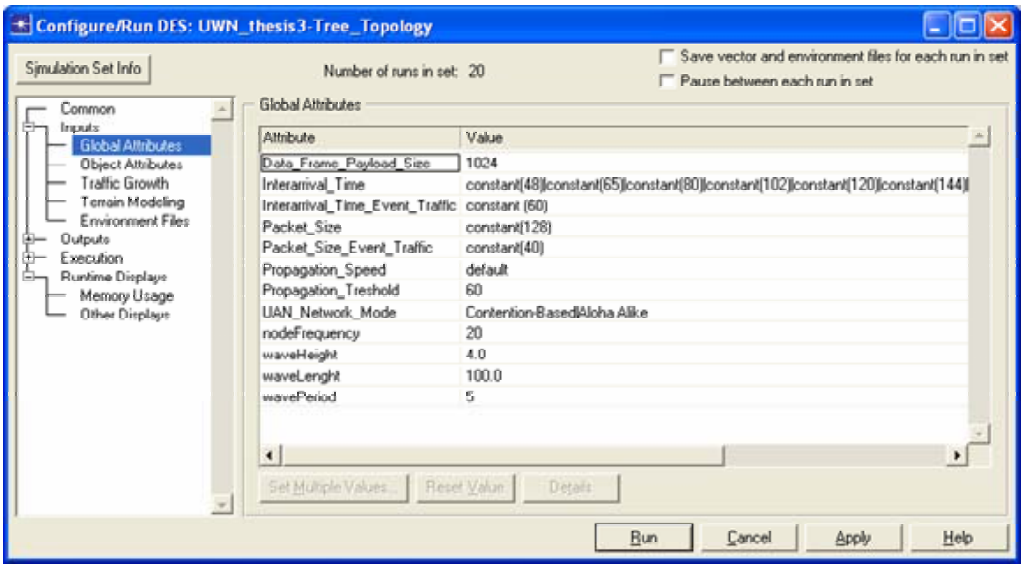

Figure 38. Global Attributes Settings.

Each simulation is composed of 20 different combinations of packet generation and network mode. The details are shown in Appendix B.

3. Performance

A typical simulation run takes approximately three and a half seconds per simulated hour, as showed in Figure 39. This achieves the thesis goal of obtaining a lightweight propagation loss algorithm.

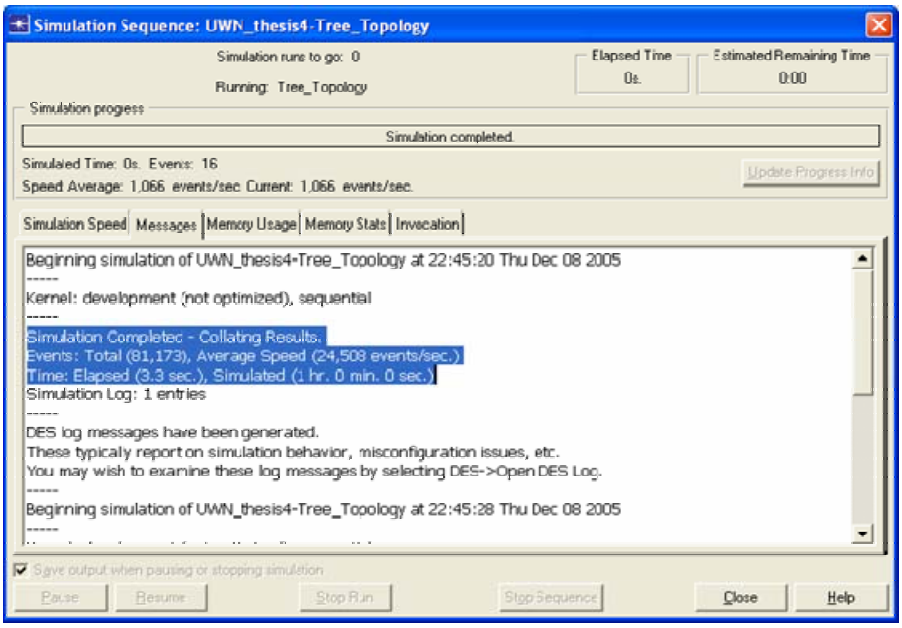

Figure 39. Screen Shot that Illustrates Simulation Performance.

4. Results from One Simulation Run

In his work, Coelho obtained the results shown in Figure 40. from a single simulation run with the random seed set to 128. It can be seen that the Aloha-like protocol performs better than the collision avoidance scheme.

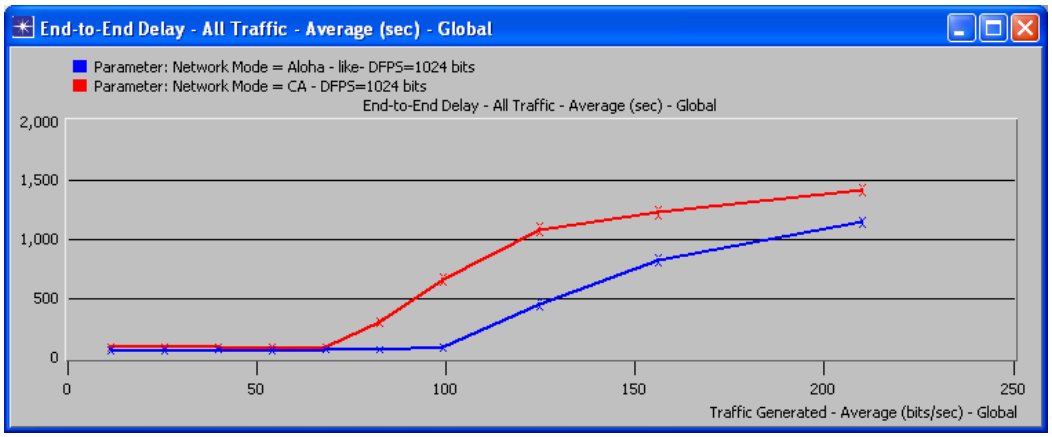

Figure 40. Tree Topology - Load vs. End-to-End Delay under Coelho's Original Model [Coelho 2005], DFPS Represents the Frame Size.

In our preliminary exploration, we performed a simulation run with the same random seed. We also obtained similar results, as shown in Figure 41. where we can also see that the Aloha-like protocol outperforms the collision avoidance protocol.

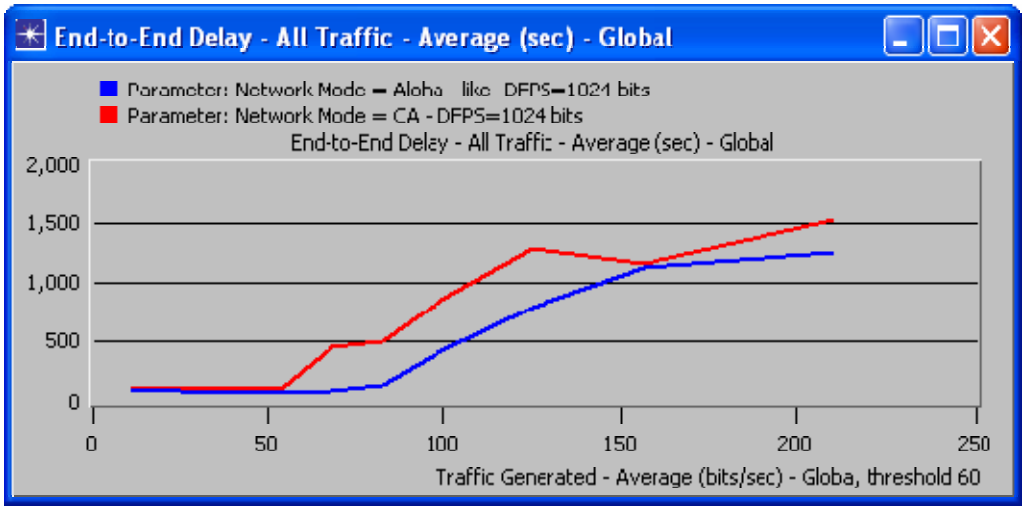

Figure 41. Tree Topology - Load vs. End-to-End Delayunder the New Algorithm with Loss Threshold Set to 60 dB.

We went further and by changing the Attributes Settings to vary the threshold value to see how it affected the simulation. The results with 50 dB thresholds are shown in Figure 42.

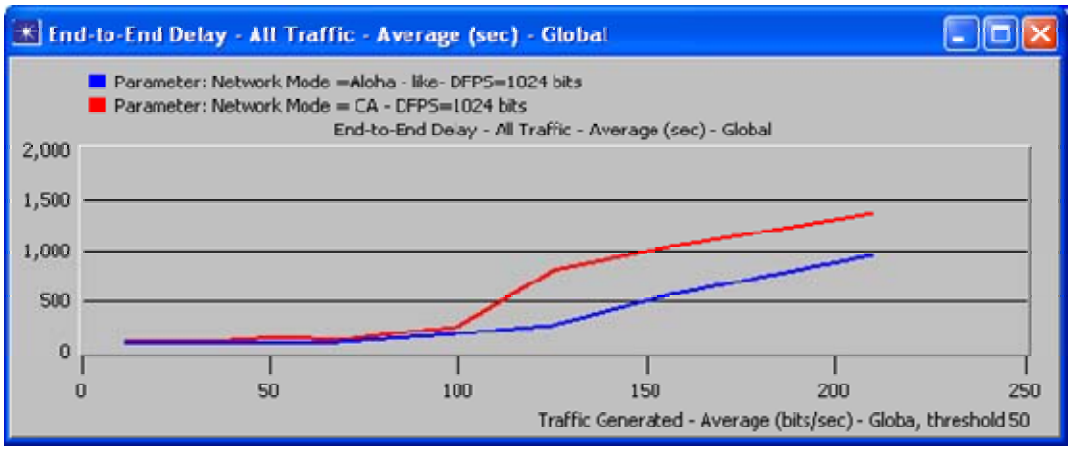

Figure 42. Tree Topology - Load vs. End-to-End Delay under the New algorithm with Loss Threshold Set to 50 dB.

When the loss threshold was lowered to 49 dB, with a transmission rate of 10 kHz we found that only the Aloha-like protocol was able to deliver traffic. We believe this is due to the fact that the lowering of the loss threshold caused transmission failures to increase, which impacted the CA scheme much more because CA requires a two-way handshake and the propagation path is not symmetric with respect to signal loss under the new model.

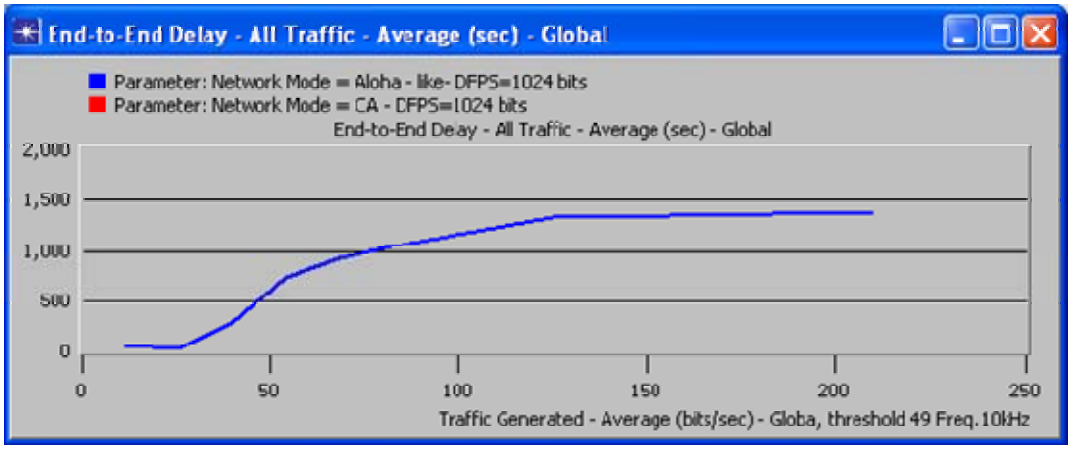

Figure 43. Tree Topology - Load vs. End-to-End Delay Loss Threshold Set to 49 dB Frequency 10 kHz.

When the transmission frequency was increased to 20 kHz, with the same loss threshold of 49 dB, both CA and the Aloha like protocols failed to delivery any frames. This is understandable because there is more loss associated with a higher frequency. We decided to explore a little further to determine the reason for this behavior. We found that as shown in Figure 37. that in Relay Node 11, the queue under the Aloha-like scheme was growing according to the load, but the one for CA was hardly active. Either this node was retransmitting the CA traffic very efficiently (which is not likely) or not receiving enough data from the sensors to fill the queue (which is more likely).

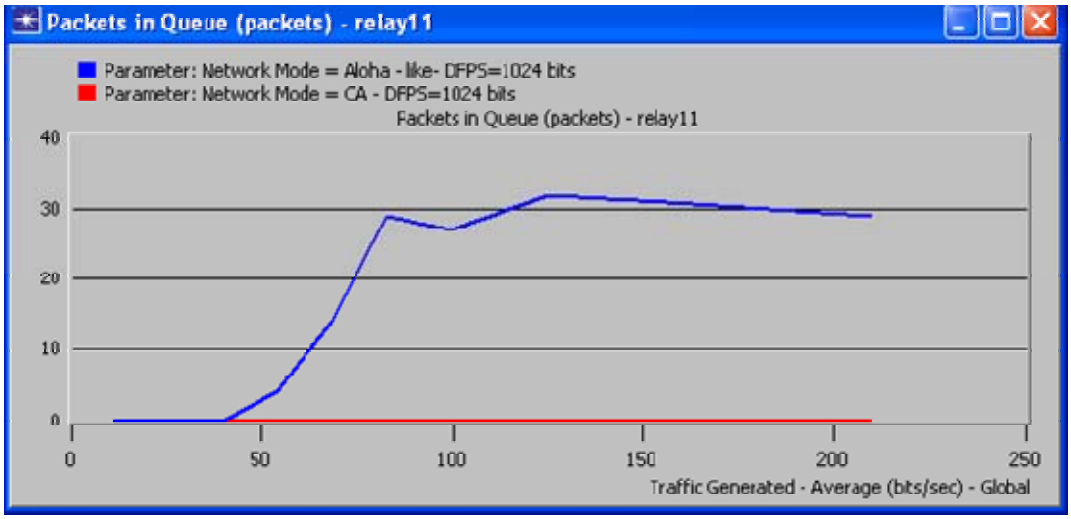

Figure 44. Relay 11 Queue Size vs. Load.

Figure 45. shows that the traffic queue of Relay Node 6 had no activity at all for CA and very little for the Aloha-like protocol. This indicates that the gateway did not receive CA traffic through this node.

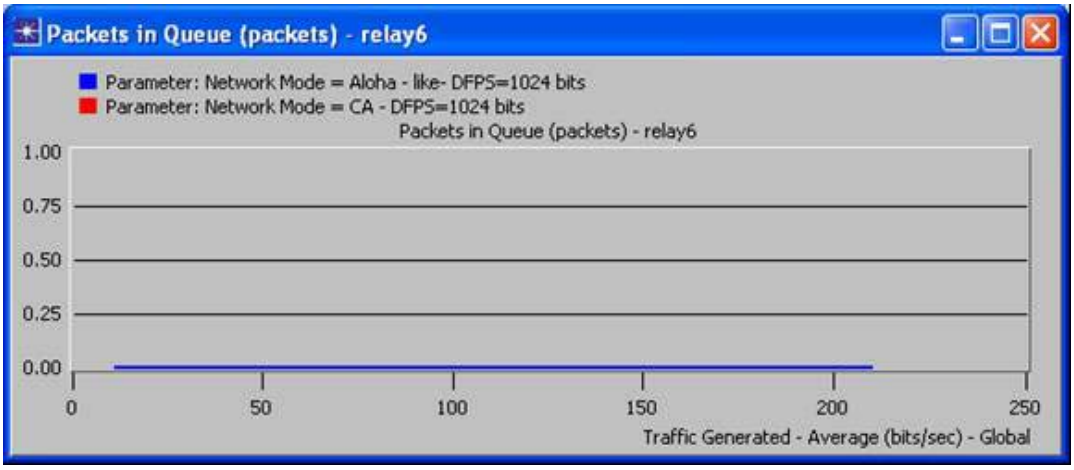

Figure 45. Relay 6 Queue Size vs. Load.

The same behavior is observed for the Relay Node 9 as shown Figure 46. Since these two relays carry all the traffic to the gateway, there was no chance that any traffic will get to the gateway under CA.

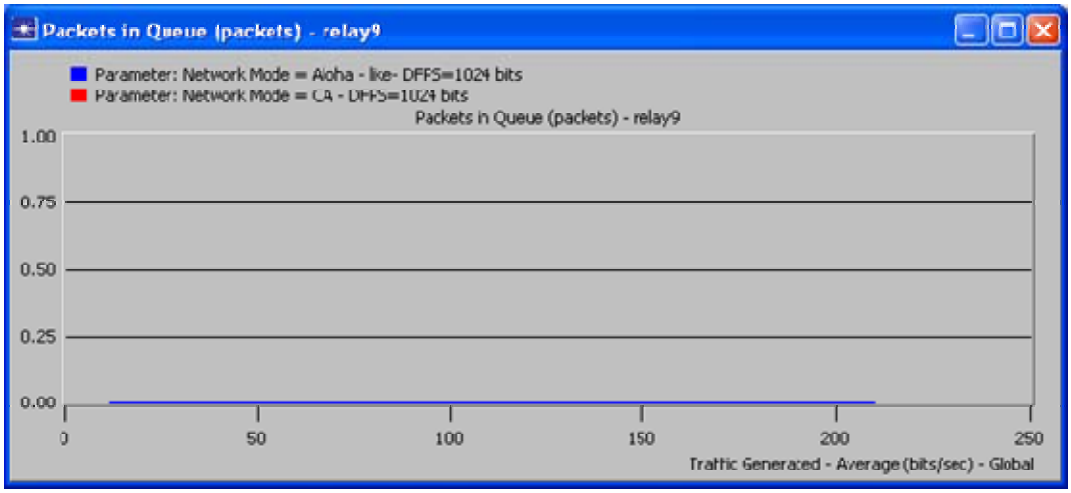

Figure 46. Relay 9 Queue Size vs. Load.

During this experiment, the Aloha-like traffic produced significantly more collisions that the CA traffic, as shown in Figure 47. This is the expected behavior of the CA and Aloha-like protocols and indicates that the problem is not protocol related.

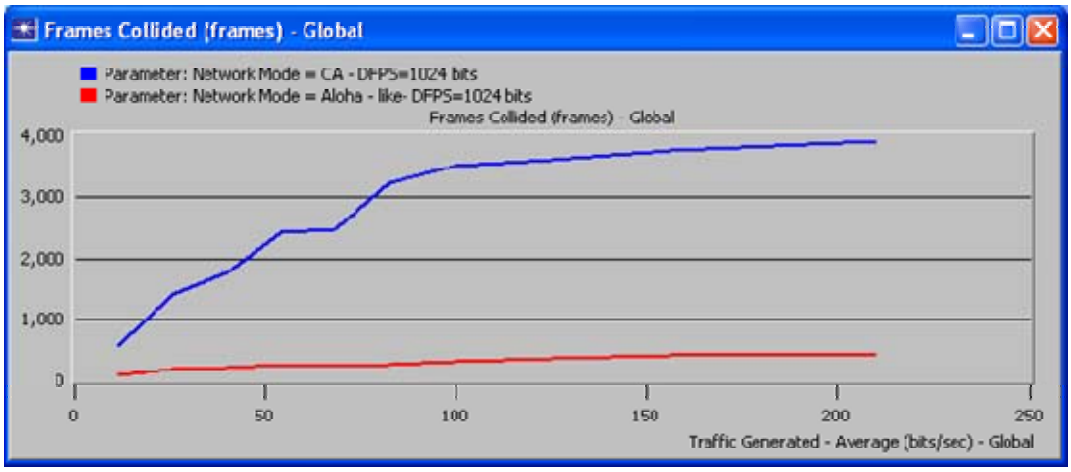

Figure 47. Collisions vs. Load.

The Aloha-like protocol succeeded in delivering messages from sensors to the gateway during this experiment but in real life the sender will have difficulty to obtain a confirmation of data receipt from the receiver if such confirmation is required because the propagation path is not symmetric with respect to signal loss. Also, it should be noted that, although traffic could reach the gateway under the Aloha like protocol, the data rate is very small, less than four bits per second as shown in Figure 48.

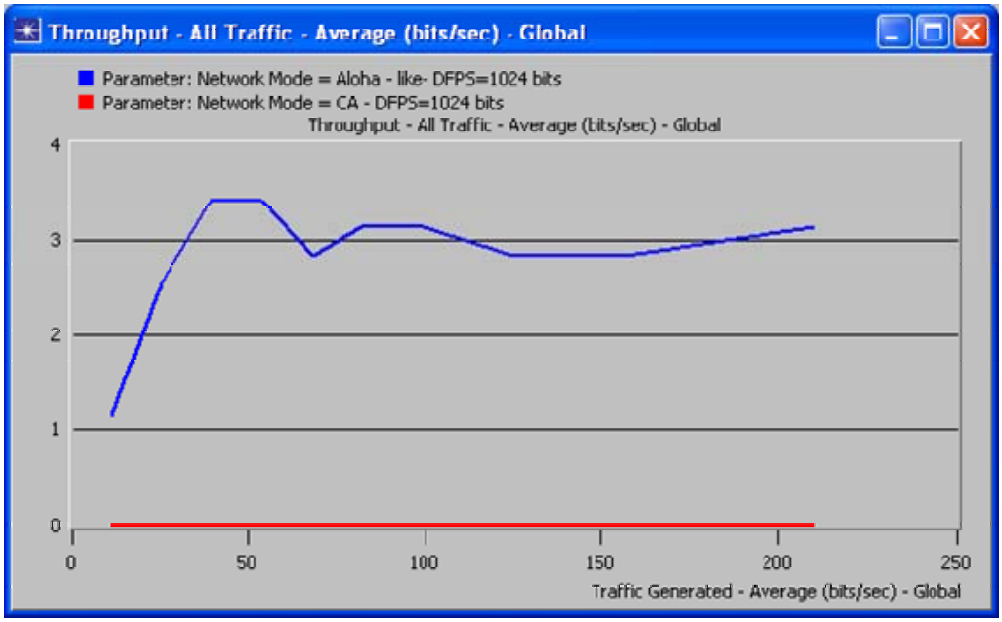

Figure 48. Throughput vs. Load.

5. Average Performance Results

To avoid having a result that is not statistically sound, in accordance with the central limit theorem, we performed a large number of simulation runs using different random seeds and then derived the mean performance. The mean should observe a normal, or close to normal, distribution. The approximation to normal depends on the sample size. The specific random seeds used are listed in Appendix B.

With two MAC protocols, 10 different traffic load factors and 30 runs for each load factor, we had to run 600 simulations for each network setting, the results are as follows.

For the setting where the loss threshold is 65 dB and transmission frequency is 10 kilohertz, we observe that the Aloha-like protocol performs almost 50% better than the CA. This is in accordance with the results obtained by Coelho.

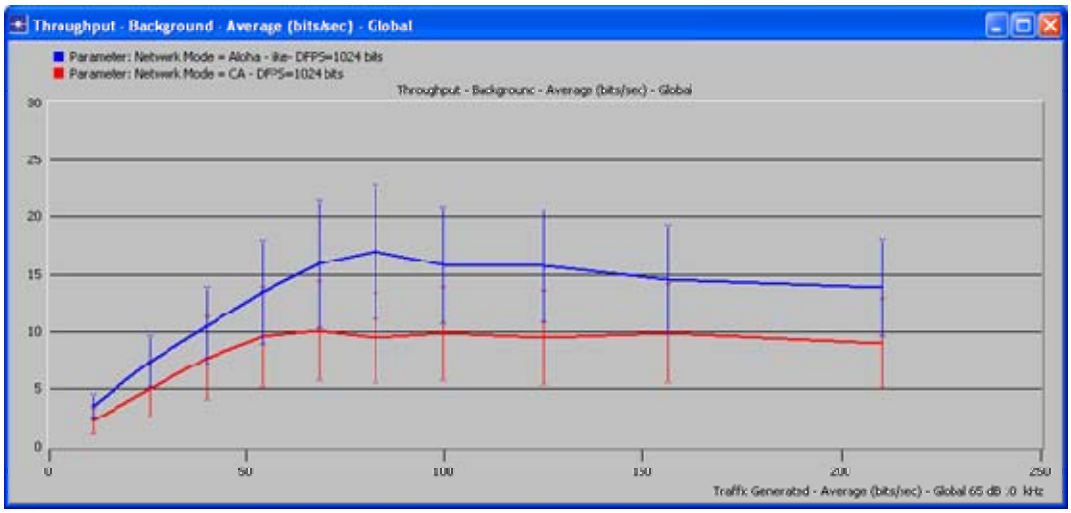

Figure 49. Throughput vs. Load Loss Threshold = 65 dB; Transmission Frequency = 10 kHz.

We then changed the transmission frequency to 20 kilohertz we found, as shown in Figure 50. , that the Aloha-like protocol outperforms the CA.

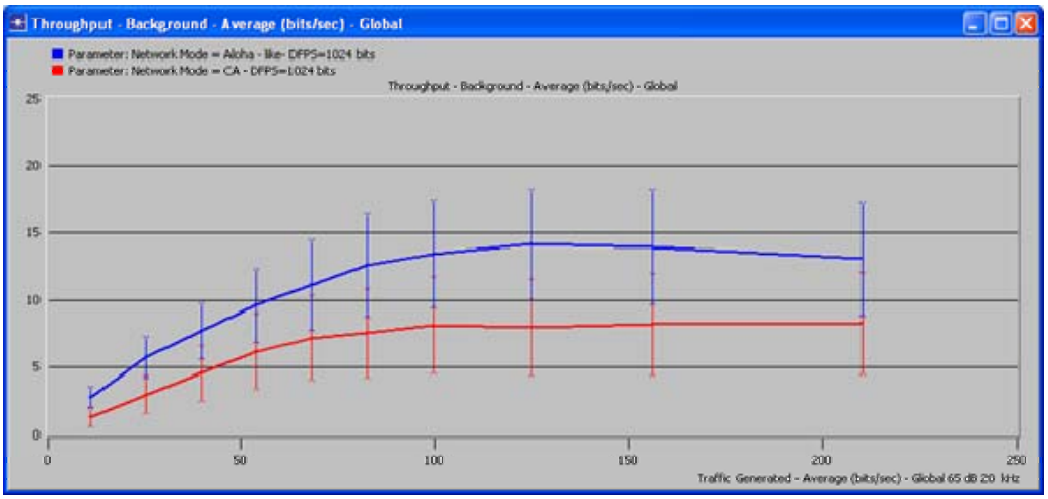

Figure 50. Throughput vs. Load Loss Threshold = 65 dB Transmission Frequency = 20 kHz.

More interesting are the findings when we analyze the end-to-end delay at 65 dB of threshold, illustrated in Figure 51. The CA protocol outperforms the Aloha-like protocol. This is unexpected as it seems odd for a protocol to have better end-to-end delay and worse throughput. A possible explanation for this is the way the statistics are collected: by calculating the end-to-end delay only for those packets that arrive at the destination, the number of CA packets that travel the whole path are less than the Aloha-like. Thus, Aloha-like has better throughput but more end-to-end delay.

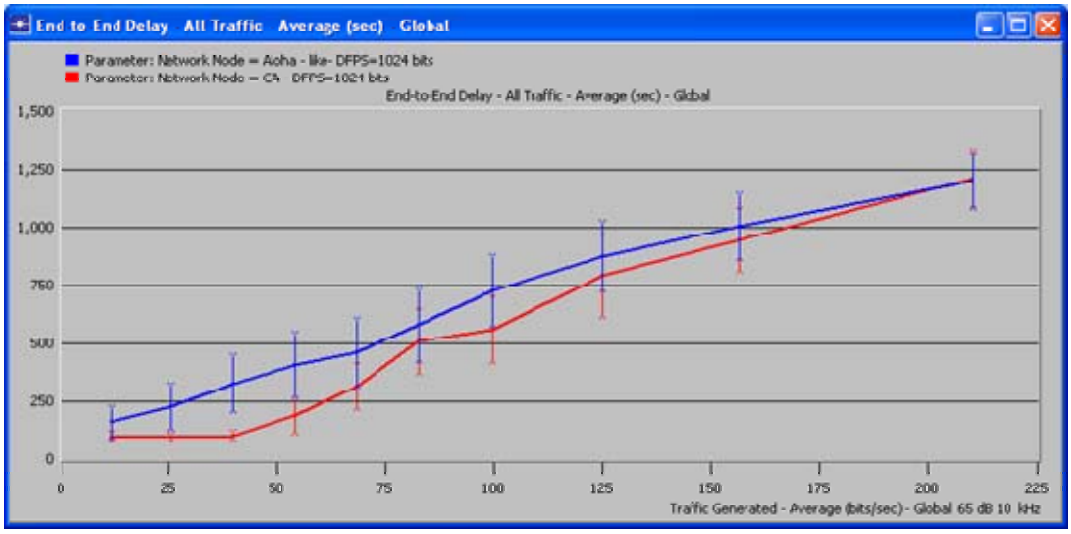

Figure 51. End to End Delay vs. Load Loss Threshold = 65 dB Transmission Frequency = 10 kHz.

We then increased the threshold to 80 dB and observed, as in Figure 52. , that the Aloha-like protocol still had better performance than the CA protocol.

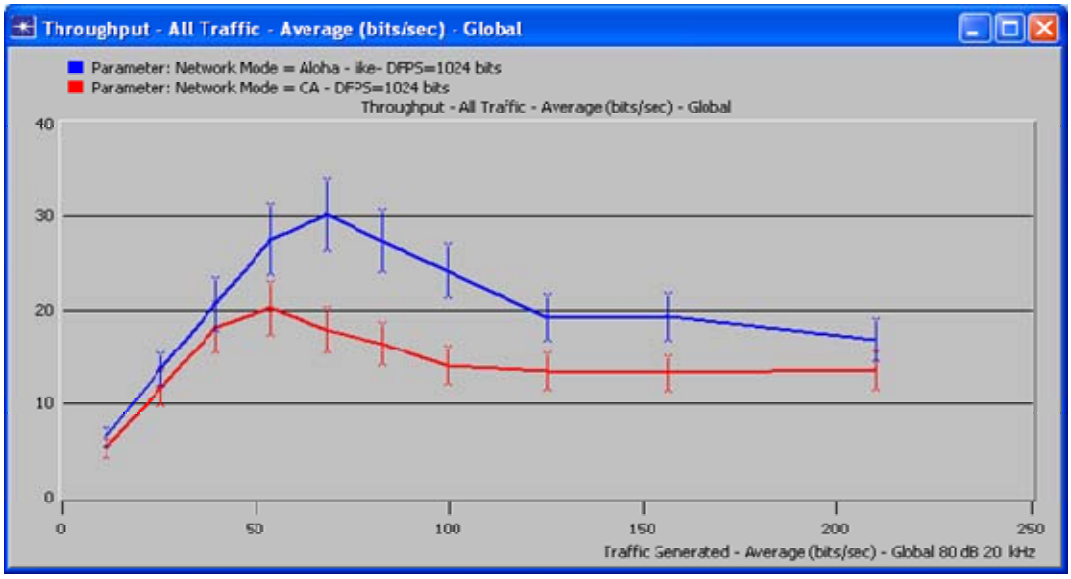

Figure 52. Throughput vs. Load Loss Threshold = 80 dB Transmission Frequency = 20 kHz.

We then analyzed the end-to-end delay at 80 dB and 20 kilohertz. From Figure 53. we can see that CA packets take more time to reach the destination on average. The interesting part of this graph is that at lower load the Aloha-like protocol has more end-to-end delay. This may be due to the number of collisions that the protocol produces, which is always greater than that for the CA protocol, as shown in Figure 54.

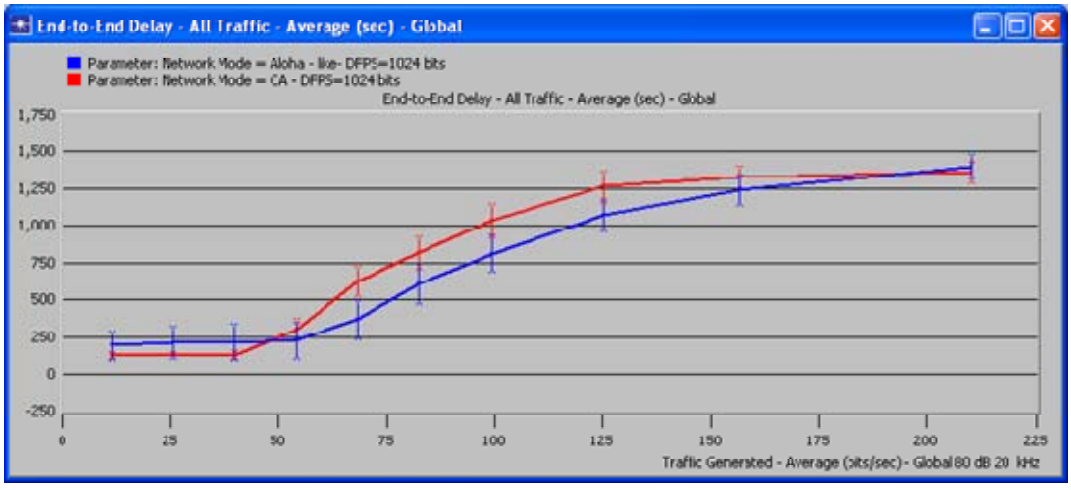

Figure 53. End to End Delay vs. Load Loss Threshold = 80 dB Transmission Frequency = 20 kHz.

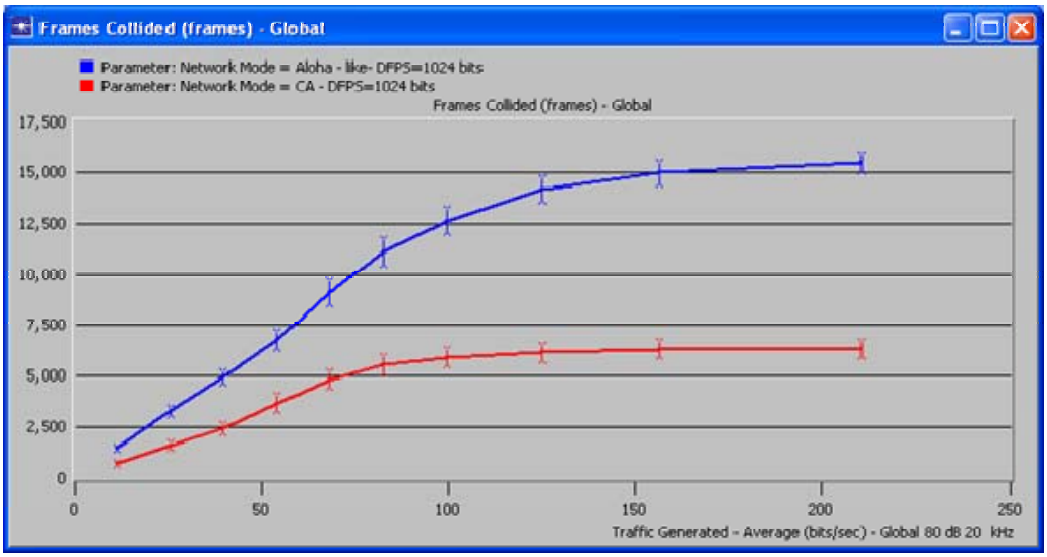

Figure 54. Collisions vs. Load Loss Threshold = 80 dB Transmission Frequency = 20 kHz.

C. CHAPTER CONCLUSION

The incorporation of this algorithm in the OpNet simulation produces very interesting results, without introducing significant overhead. The intriguing results may be due to the inherent randomness of the propagation loss algorithm. Such findings as having more throughput and more end-to-end delay at lower traffic loads may warrant a closer look to determine the causes of such behavior. We also noticed that the values are very close and the confident intervals overlap. The end-to-end delay differential may not be as significant as thought from the Coelho experiments. We also observed the direct relation of propagation loss to transmission frequency, but there are more parameters that can be changed, like depth or the wave period, which may influence the outcome. These values were maintained constant throughout this series of simulations. More experiments are needed to develop a better understanding of the propagation algorithm's implications on those two protocols.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS

A. CONCLUSION

This thesis presents a method for creating an efficient algorithm to compute acoustic signal propagation loss, by means of a statistical approximation of a chosen physical model of acoustic propagation. The method was applied to a well-known physical model (MMPE), resulting in a viable approximation algorithm. The algorithm has considerably less overhead than the original physical model (MMPE) while it is accurate enough to be incorporated into an underwater network simulation. One hour of simulation using an OpNet model embedded with the algorithm takes approximately 340 times less time than a single MMPE area calculation. Three major conclusions can be drawn from this effort:

- Due to the exceedingly high computational complexity, relative to today's CPU capability, a direct implementation of a physical model of propagation loss such as MMPE within a network simulation is not practical at this point in time.
- It is relatively straightforward to incorporate the class of approximation algorithms obtained with the statistical method into existing UAN simulation programs.
- It is important to upgrade existing UAN simulation programs with a more realistic acoustical propagation model. The OpNet experimental results presented in this thesis indicate that the performance of an UAN protocol may be significantly impacted by the introduction of a more advanced propagation model at the physical layer.

B. RECOMMENDATIONS

- The statistical method is promising as it is easy to use and able to create computationally-efficient algorithms. It shall worth the effort to explore it further. While there is no established ways of verifying the fidelity of an algorithm created using the statistical method, it can be expected that the fidelity of such algorithms will improve over time as they are used in more UAN studies.
- The existing conclusions regarding UAN protocols should be re-examined using more advanced acoustic propagation models. At the least, the wave effect should be explicitly modeled as it may cause volatile and possibly periodic (and thus frequent) fluctuations in the quality of the underwater.

transmission channel. Such fluctuations in channel quality have so far not received adequate attention in the underwater acoustic communication community.

C. FUTURE WORK

- Develop a methodology to verify the fidelity of an algorithm produced using the statistical method. The hypothesis of a periodic wave effect needs to be tested. A possible approach is to use data from real world experiments.
- Demonstrate the generality of the statistical method by applying it to different physical models of underwater acoustic propagation, including those which explicitly model the wave effect and random errors.
- Extend the statistical method so that it can create algorithms that model additional physical properties such as the sea bottom profile and the sound propagation speed profile.

APPENDIX A. PHYSICAL LAYER PROCESS MODEL MODIFICATIONS C-LANGUAGE CODE (OPNET)

```
================================================================================
                                Function Block
================================================================================
/* ############################################################################
Utility function for populate the list of reachable nodes
*/
static void defineReachableNodes()
{
        int idx;
        //int idx1;
        int nrNodes;
        Objid externNode;
        double myXposition;
        double myYposition;
        double myDepth;
        double externXposition;
        double externYposition;
        double distance;
        double externdepth;
        double propagationLoss;

        UanT_Reachable_Neighbors* reachableNeighbor;

        FIN (defineReachableNodes());

        // Our geographical position
        op_ima_obj_attr_get (myNodeObjectID, "x position", &myXposition);
        op_ima_obj_attr_get (myNodeObjectID, "y position", &myYposition);

        //LD:to assing the depth to the state variable
        op_ima_obj_attr_get (myNodeObjectID, "MAC Layer.depth", &myDepth);

        // Number of nodes in the network
        nrNodes = op_topo_child_count (myNetworkObjectID, OPC_OBJMTYPE_NODE);

        /* Loop through all the nodes in the network and check if they are within
           our range */
        for (idx = 0; idx < nrNodes; idx++)
        {
                // Get the idx node in the network
                externNode = op_topo_child (myNetworkObjectID, OPC_OBJMTYPE_NODE, idx);

                // Do not perform the algorithm for my own node
                if (externNode != myNodeObjectID)
                {
                        // Get the geographical position of the idx neighbor node
                        op_ima_obj_attr_get (externNode, "x position", &externXposition);
                        op_ima_obj_attr_get (externNode, "y position", &externYposition);


                        // LD: Get the depth of the idx neighbor node

                        op_ima_obj_attr_get (externNode, "MAC Layer.depth", &externdepth);


                        // LD:Calculate the distance using depth too

                        distance = getDistance (myXposition, myYposition, myDepth,
                                                externXposition, externYposition,
externdepth);

                        //LD:compute the PL function
                        propagationLoss = propLoss(
nodeFrequency,myDepth,distance,externdepth,wavePeriod,waveHeight);
```

```
                        //LD: add node to the list if it has a loss lessor iqual than the
threshold


                        if(propagationLoss <= PropagationTreshold)

                           {
                                // Allocate memory for the new list element
                                reachableNeighbor = (UanT_Reachable_Neighbors*)
                                       op_prg_mem_alloc (sizeof
(UanT_Reachable_Neighbors));

                                // Add node's name
                                op_ima_obj_attr_get (externNode, "name",
                                       reachableNeighbor->name);

                                // Add node's propagation delay
                                reachableNeighbor->propagationDelay =
                                       distance / propagationSpeed;

                                // LD:Add node's signalStrenght
                                reachableNeighbor->signalStrenght =
                                       propagationLoss;


                                // Add this element to the list
                                op_prg_list_insert (reachableNeighborsList,
reachableNeighbor,
                                       OPC_LISTPOS_TAIL);
                        }

                }

        } // end for

        FOUT;
}



/* ###########################################################################
 LD:compute the m() function
*/

   static double propLoss ( double frequency,double Sourcedepth, double Distance,
        double receiverdepth, double waveP,double waveH)
   {

    double pLoss;
        double wavePL;
        double plW;
        double plr;






        FIN (propLoss (frequency,Sourcedepth,Distance,receiverdepth,waveP,waveH));


        //LD: Calculate propagation loss


                        pLoss = getPloss(frequency,Sourcedepth,Distance,receiverdepth);
```

```
        //LD:calculate the effect due to wave movement

                        wavePL = getWavePL ( waveLenght,op_sim_time(),waveP, waveH,
receiverdepth );

        //LD:adds the wave feect to the Propagation loss

                        plW = addWavePL(pLoss,wavePL);

   //LD:gets the random component

                        plr = getRandompl(Distance);

        //LD:returns the propagation loss
        FRET (plW+plr);

}
```

```
/* ###########################################################################
LD: Accept the geographical position of two nodes in meters and calculates the
direct distance between them using depth
*/
static double getDistance (double myXpos, double myYpos,double myZpos, double otherXpos,
                                double otherYpos,double otherZpos)
{
        FIN (getDistance (myXpos, myYpos,myZpos, otherXpos, otherYpos, otherZpos));

        FRET (sqrt( (pow((myXpos - otherXpos), 2.0) + pow((myYpos - otherYpos), 2.0))+
                  pow((myZpos - otherZpos), 2.0)  ));
}


/* ###########################################################################
LD:Accept the distance between two nodes in meters and calculates the
propagation loss between them
*/
static double getPloss ( double f,double dA,double s,
        double dB)
{

//LD:coeficients obtained by regresion

double A0 = 27.3111;
double A1  = -113.469;
double A5  = 0.0274476;
double A6  = 0.00495431;
double A7  = -19.296;
double A8  = 0.123063;
double A9  = -0.102525;
double A10 = 0.526268;
double V5  = 0;
double f2=0;
```

```
        FIN (getPloss (f, dA,s,dB));

//LD:precomputes f2

    f2 = pow(f,2);


//LD:obtains the propagation loss from the regresion model
        V5 = log (fabs(    ( pow( (s/0.914),A0 )*    pow( (pow( (dA-dB),2 )),A10  )*
pow(dA,A9)*pow(s,A7) ) / ( pow( (s * dB),(10*A5)) )  )    )+ (f2*( A1/(1+f2)+
40/(4100+f2)+0.00275 )+0.003)*s/914+A6 * dB+ A8 * s;


//printf("\n - pLoss %f \n", V5);

//LD:returns the propagation loss due to regresion model

        FRET (V5);
}


/* ###########################################################################
LD: compute the wave effect by Accepting the wave characteristics and calculates the wave
efect
*/
static double getWavePL ( double lw , double time,double period,double height, double
receiverDepth )

{

double aW;
double rW;
double vM;
double H;
double rn;

     FIN (getWavePL (lw, time, period, height, receiverDepth ));

        //LD:to avoid divide by 0 at time 0
    if (time == 0)
              time= time +1;

   //LD: calculates the advance of the wave (coeficient to multiply times 2PI

        aW = (time/period-((int)(time/period)));

         //   printf("\n - aw %f \n", aW);


   //LD:computes the ratio of the wave

        rW = (2*receiverDepth)/lw;

   // LD:to avoid error

   if( rW > 1)
         {
           rW = 1;
         }
     //printf("\n - rw %f ", rW);

   //LD:calculates the Height of the wave movement
    H = height*(1-rW)/0.5;


        //LD:calculates the vertical movement induced to the node

    vM = H*fabs(sin(2*PI*aW))/2;
```

```
        // LD:to assure this function is called one time
       if (oneTime == OPC_FALSE)
                {

        //LD: set the variable to true
                oneTime = OPC_TRUE;

      //LD: loads the random into the structure

                waveEffect.Rn1 = op_dist_outcome(wave_dist);
                waveEffect.Rn2 = op_dist_outcome(wave_dist);
                waveEffect.Rn3 = op_dist_outcome(wave_dist);
                waveEffect.Rn4 = op_dist_outcome(wave_dist);
                waveEffect.Rn5 = op_dist_outcome(wave_dist);
                waveEffect.Rn6 = op_dist_outcome(wave_dist);
                waveEffect.Rn7 = op_dist_outcome(wave_dist);
                waveEffect.Rn8 = op_dist_outcome(wave_dist);

                //LD: unloads the wave distribution , not longer needed, asures one time
                op_dist_unload (wave_dist);

                }


        //LD: series of if statements  to determine the position of the movement and
assign the correspondent value

        if (aW >= (7.0/8.0))
                {

                rn = waveEffect.Rn8;

                }
            else
            if (aW >= (6.0/8.0))
                   {
                    rn = waveEffect.Rn7;
                   }
                   else
               if (aW >= (5.0/8.0))
                       {
                        rn = waveEffect.Rn6;
                       }
                       else
                  if (aW >= (4.0/8.0))
                          {
                           rn = waveEffect.Rn5;
                          }
                          else
                     if (aW >= (3.0/8.0))
                             {
                              rn = waveEffect.Rn4;
                             }
                             else
                        if (aW >= (2.0/8.0))
                                {
                                 rn = waveEffect.Rn3;
                                }
                                else
                           if (aW >= (1.0/8.0))
                                   {
                                    rn = waveEffect.Rn2;
                                   }
                                   else
                                   {
                                    rn = waveEffect.Rn1;
                                   }



        //LD: return the wave effect value
```

```
        FRET (vM*rn);
}


/* ###########################################################################
LD:Accept the Accept the  wave efect and the propagation loss and adds them
*/
static double addWavePL(double wavePL,double pLoss)
{
        FIN (addWavePL(wavePL, pLoss));

        FRET (wavePL+pLoss);


}
/* ###########################################################################
LD:Accepts the distance and calculates the signal random error

*/
static double getRandompl(double distance)
{
        double randError;
        //LD:set to 1000 because the regresion is up to that ( it will increase more than
20 db after that)
        double rangeMax = 1000.0;


FIN (getRandompl(distance));

        /*LD:calculates the random error scaled so it can be  20 db at maximum range, and
the distribution
        is also scaled to be one at three standard deviations */

                randError =op_dist_outcome (randomError_dist)*(distance/rangeMax)*(20/3);

                printf("\n - plW Random error %f \n", randError);



        FRET (randError);
}
/* ###########################################################################
LD: determines if a packet causes a collision by obtaining the signal loss
*/
static Boolean disturbing(Packet* frame)
{

UanT_Frame_FieldsP*     collisionHeader;
Objid macLayer2;
double *aver = NULL;
Boolean check;


FIN (disturbing(frame));

    //LD:assures the return variable is false
   check = OPC_FALSE;

        //LD: obtains the header from the frame
        op_pk_fd_get (frame, 0, &collisionHeader);

   //LD:obtains the pointer to the mac layer object
    macLayer2 = op_id_from_name (myNodeObjectID, OPC_OBJTYPE_QUEUE, MAC_LAYER);

        //printf("\n -ya llegue %f \n");

        // LD:Retrieve the attribute needed for the packet strength
        aver = (double *)op_ima_obj_svar_get (macLayer2, "packetInStrenght");
```

```
   //LD:obtains the packet in process strength

    ppStrenght= * aver;


         printf("\n -previous packet loss %f \n",ppStrenght);

// LD:if this is the first packet accept
          if(ppStrenght == 1)
                check = OPC_TRUE;

  // LD:if the packet loss is bigger than the threshold ignore the packet

          //printf("\n -PropagationTreshold packet loss %f \n",PropagationTreshold);

       if(collisionHeader->strenghOfSignal > PropagationTreshold)
                  check = OPC_FALSE;



       //op_pk_destroy (frame);

       //LD:return the Boolean
       FRET (check);


}

================================================================================

                    Enter Execs for the forced state "send"

================================================================================

/* This state process the reception of the frame from our MAC layer, and

   delivers it to all reachable nodes */


// Retrieving the frame from the stream used by our link layer to sent us frames

currentFrame = op_pk_get(INPUT_STREAM_FROM_MAC_LAYER);




/* We do not need to signal the beginning of transmission, because the MAC layer

assumes the beginning of transmission as the moment it sends the frame to us.

However we need to signalling the end of transmission with the correct

transmission delay */

op_intrpt_schedule_remote (op_sim_time() +

       TXTIME (op_pk_total_size_get (currentFrame)), UwnE_Transmitter_Off,

       myMacLayer);
```

```
//LD: obtain the header to modify the strengh of the signal acording to the node
op_pk_fd_get (currentFrame, 0, &dummySignal);

/* Loop through the reachable neighbors list and send the begin and end of
reception, and the frame itself */

for (idx = 0; idx < op_prg_list_size (reachableNeighborsList); idx++)
{
        // Handle to the idx element of the list
        reachableNeighbor = (UanT_Reachable_Neighbors*)
                op_prg_list_access (reachableNeighborsList, idx);
        // Handle to the idx neighbor
        remoteNode = op_id_from_name(op_topo_parent(op_topo_parent(op_id_self())),
                OPC_OBJTYPE_NODE_FIX, reachableNeighbor->name);
//      LD: assign the signal strenght to the packet
        dummySignal->strenghOfSignal = reachableNeighbor->signalStrenght;

            //  printf("\n -signal loss  %f \n",dummySignal->strenghOfSignal);

// LD: re-incorporate the dummySignal header to the current frame
        op_pk_fd_set (currentFrame, 0, OPC_FIELD_TYPE_STRUCT, dummySignal, 0,
op_prg_mem_copy_create, op_prg_mem_free, sizeof (UanT_Frame_FieldsP));
        //printf("\n -ya frame type %f \n",dummySignal->frameType);
        // Print to debug mode
        printf("\n#%s# The physical layer is sending begin of reception, the frame,""and
end of reception events to %s", myName,  reachableNeighbor->name);

        // Handle to the physical layer module of the idx neighbor
        remotePhysicalLayer = op_id_from_name(remoteNode, OPC_OBJTYPE_PROC,
                PHYSICAL_LAYER);

        /* Scheduling the beginning of reception for the idx neighbor with the
           correct propagation delay */
        op_intrpt_schedule_remote (op_sim_time() +
```

```
                reachableNeighbor->propagationDelay, UwnE_Receiver_On,
                remotePhysicalLayer);

        /* Scheduling the deliver of the frame to the idx neighbor, but before copy
           the frame. A sent frame changes owner, and without that we could only
           sent one */
        copyFrame = op_pk_copy(currentFrame);
    op_pk_deliver_delayed(copyFrame, remotePhysicalLayer,
                INPUT_STREAM_EXTERNAL_NODES,
                TXTIME (op_pk_total_size_get (currentFrame)) +
                reachableNeighbor->propagationDelay);

        /* Schedule the end of reception for the same time that the frame was sent.
           The time added is 1 psec to guarantee the end of reception after the
           actual frame reception */
        op_intrpt_schedule_remote (op_sim_time() +
                TXTIME (op_pk_total_size_get (currentFrame)) +
                reachableNeighbor->propagationDelay + PRECISION_RECOVERY,
                UwnE_Receiver_Off, remotePhysicalLayer);

}

// After sent a copy to all nodes, destroy the frame received from the MAC layer
op_pk_destroy(currentFrame);
--------------------------------------------------------------------------------


================================================================================
                    Enter Execs for the forced state "receive"
================================================================================
/* This state process the reception of the frame from other nodes, and deliver
it to our MAC layer */
```

```
/* Retrieving the frame, sent by some node, from the stream emulating the physical medium
*/
currentFrame = op_pk_get(INPUT_STREAM_EXTERNAL_NODES);


// Print for debugging mode
printf("\n#%s - The physical layer received a frame and is sending it to the "
           " MAC layer", myName);


/*LD: this code is muted it works but is replaced by an algorith at the sender side
it is to determine if the packet is strong enough to cause a collision, it is a good
place to
implement a more elaborated  collision detection schema */
//LD:copy the frame to be analized
//LD:copyFrameToo = op_pk_copy(currentFrame);


// LD:Send the packet to the MAC layer if signal is strong enough
//LD:if(disturbing(copyFrameToo))
op_pk_send(currentFrame, OUTPUT_STREAM_TO_MAC_LAYER);
//LD:else
// LD:Print for debugging mode
//LD:printf("\n#%s - The physical layer rejected a frame and is ignore it  "
//         "  ", myName);
//LD:op_pk_destroy (copyFrameToo);
```

APPENDIX B. OPNET SIMULATION SET

```
Simulation invocation:

  op_runsim

    -opnet_user_home C:\Documents and Settings\student1

    -mem_optimize true

    -net_name UWN_thesis3-Tree_Topology

    -noprompt

    -c

    -handle_exception true

    -ef UWN_thesis3-Tree_Topology


Associated actions:

  The scalar file will be deleted before execution.


Content of "UWN_thesis3-Tree_Topology.ef" file:


Preferences common to all sets:

  _locale : "C"

  anim_view : "false"

  comp_trace_info : "true"

  default_site_position_cache_time_granularity : "0"

  diag_enable : "true"

  duration : "3600"

  endsim_memstats : "false"
```

```
  kernel_type : "development"
  log_endsim_perf : "true"
  log_file : "UWN_thesis3-Tree_Topology"
  max_log_entries : "200"
  num_collect_values : "100"
  os_file                                                    :
"scalar_thesis_tree_P1024_F1024_dephprueba60f20mu"
  ov_file : "UWN_thesis3-Tree_Topology"
  parallel_sim.event_execution_time_window : "0"
  parallel_sim.mem_balance_interval : "100000"
  parallel_sim.num_processors : "1"
  probe : "UWN_thesis3-Tree_Topology"
  probe_start_time : "0"
  realtime_ratio : "0"
  rec_err_suppress : "false"
  seed : "128"
  sim_packet_sharing : "conservative"
  sim_time_quantum : "0"
  tmm_simulate : "false"
  update_interval : "500000"
  verbose_cell_size_report : "false"
  verbose_event_report : "false"
  verbose_event_timing_report : "false"
  verbose_load : "true"
  verbose_packet_report : "false"
```

```
  verbose_parallel_speedup_report : "false"
  verbose_sim : "true"
  warning_suppress : "false"

Expected runs based on varying elements:
Simulation run #1:
  Interarrival_Time: constant(48)
  UAN_Network_Mode: Contention-Based

Simulation run #2:
  Interarrival_Time: constant(65)
  UAN_Network_Mode: Contention-Based

Simulation run #3:
  Interarrival_Time: constant(80)
  UAN_Network_Mode: Contention-Based

Simulation run #4:
  Interarrival_Time: constant(102)
  UAN_Network_Mode: Contention-Based

Simulation run #5:
  Interarrival_Time: constant(120)
  UAN_Network_Mode: Contention-Based
```

```
Simulation run #6:
  Interarrival_Time: constant(144)
  UAN_Network_Mode: Contention-Based

Simulation run #7:
  Interarrival_Time: constant(180)
  UAN_Network_Mode: Contention-Based

Simulation run #8:
  Interarrival_Time: constant(240)
  UAN_Network_Mode: Contention-Based

Simulation run #9:
  Interarrival_Time: constant(360)
  UAN_Network_Mode: Contention-Based

Simulation run #10:
  Interarrival_Time: constant(720)
  UAN_Network_Mode: Contention-Based

Simulation run #11:
  Interarrival_Time: constant(48)
  UAN_Network_Mode: Aloha Alike

Simulation run #12:
```

```
  Interarrival_Time: constant(65)
  UAN_Network_Mode: Aloha Alike

Simulation run #13:
  Interarrival_Time: constant(80)
  UAN_Network_Mode: Aloha Alike

Simulation run #14:
  Interarrival_Time: constant(102)
  UAN_Network_Mode: Aloha Alike

Simulation run #15:
  Interarrival_Time: constant(120)
  UAN_Network_Mode: Aloha Alike

Simulation run #16:
  Interarrival_Time: constant(144)
  UAN_Network_Mode: Aloha Alike

Simulation run #17:
  Interarrival_Time: constant(180)
  UAN_Network_Mode: Aloha Alike

Simulation run #18:
  Interarrival_Time: constant(240)
```

```
  UAN_Network_Mode: Aloha Alike


Simulation run #19:
  Interarrival_Time: constant(360)
  UAN_Network_Mode: Aloha Alike


Simulation run #20:
  Interarrival_Time: constant(720)
  UAN_Network_Mode: Aloha Alike
```

LIST OF REFERENCES

[Bell 1962] Bell, T. G., Sonar and Submarine Detection, US Navy Underwater Sound Lab, Rep. 545, 1962.

[Coelho 2005] Coelho, Jose, "Underwater Acoustic Networks: Evaluation of the Impact of Media Access Control on Latency, in a Delay Constrained Network," Master's Thesis (MS-CS), Naval Postgraduate School, Monterey, California, March 2005.

[Dorn 1962] Van Dorn, W.**,** "Oceanography and Seamanship," Dodd, Mead, New York, 1974.

[Gibson 2005] Gibson, John, Xie, Geoffrey, Coelho, José and Diaz-Gonzalez, Leopoldo. "The Impact of Contention Resolution verses a priori Channel Allocation on Latency in a Delay Constrained Network," Pending publication by the Undersea Networking Working Group, in support of The Technical Cooperative Program (TTCP).

[Lysanov 1982] Brekhovskikh, L. and Lysanov, Yu, "Fundamentals of Ocean Acoustics," Springer-Verlag Berlin Heidelberg New York, 1982.

[OpNet 2003] OpNet Version 11.0, "Wireless Manual: Radio Transceiver Pipeline," Modeler, Documentation, OPNET Technologies, and Inc., 2003.

[Smith 1999] Smith, Kevin B., "Convergence, Stability, and Variability of Shallow Water Acoustic Predictions Using a Split-Step Fourier Parabolic Equation Model," Shallow Water Acoustic Modeling (SWAM'99) Workshop, 1999.

[Sozer 1999] Sozer, Ethem M., Stojanovic, Milica and Proakis, John G., "Design and Simulation of an Underwater Acoustic Local Area Network,**"** Northeastern University, Communications and Digital Signal Processing Center, 409 Dana Research Building, Boston, Massachusetts, 1999.

[S-PLUS 2005] S-Plus Version 7.0, Help Documentation, Insightful Corp. 2005.

[Tappert MMPE] Smith, Kevin B. and Tappert, Frederick, "Monterrey-Miami Parabolic Equation," Release Documentation.

[Urick 1983] Urick, Robert J., "Principles of Underwater Sound," 3rd Edition, Peninsula Publishing, 1983.

[Xie 2001] Xie, Geoffrey and Gibson, John, "A Network Layer Protocol for UANs to Address Propagation Delay Induced Performance Limitations," Proceedings of the MTS/IEEE Oceans Conference, November 2001.

[Yonghoon 2000] Yonghoon, Ha, "Java-Based Implementation of Monterey-Miami Parabolic Equation (MMPE) Model with Enhanced Visualization and Improved Method of Environmental Definition," Master's Thesis (MS-EA), Naval Postgraduate School, Monterey, California, December 2000.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Centro de Estudios Superiores Navales
   Mexican Navy
   Mexico City, DF
   Mexico

4. Geoffrey Xie
   Naval Postgraduate School
   Monterey, California

5. John Gibson
   Naval Postgraduate School
   Monterey, California